AD-772 614

DEVELOPMENT OF NEW PATTERN-RECOGNITION METHODS

STANFORD RESEARCH INSTITUTE

PREPARED FOR
AEROSPACE RESEARCH LABORATORIES

NOVEMBER 1973

DISTRIBUTED BY:

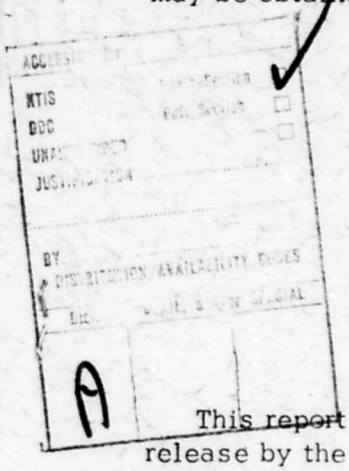
NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Organizations or individuals receiving reports via Aerospace Research Laboratories automatic mailing lists should refer to the ARL number of the report received when corresponding about change of address or cancellation. Such changes should be directed to the specific laboratory originating the report. Do not return this copy; retain or destroy.

Reports are not stocked by the Aerospace Research Laboratories. Copies may be obtained from:



National Technical Information Services
Clearinghouse
Springfield, VA 22151

This report has been reviewed and cleared for open publication and public release by the appropriate Office of Information in accordance with AFR 190-12 and DODD 5230.0. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service.

AIR FORCE/56780/4 December 1973 — 500

UNCLASSIFIED

Security Classification

AD-772614

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, California 94025

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

Development of New Pattern-Recognition Methods

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Scientific Final

5. AUTHOR(S) *(First name, middle initial, last name)*

D. J. Hall D. A. Huffman
R. O. Duda D. E. Wolf

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November 1973	234	146

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F33615-71-C-1894	
b. PROJECT NO. 7071-02-12	ARL 73-0153
c. DoD Element 61102F	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d. DoD Subelement 681304	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Aerospace Research Laboratories (LB) Wright-Patterson AFB, Ohio 45433

13. ABSTRACT

The problem in pattern recognition is to find a classification or description of the data patterns that matches or suits the data. New methods in pattern recognition are studied in relation to classical approaches using techniques of multivariate statistical analysis. The application of these techniques to specific problems in physical, engineering, behavioral, and other sciences is reviewed. The problems of improved data description and dimensionality reduction are tackled by means of clustering approaches. Several improved clustering methods are developed for general pattern recognition: a new approximate procedure for computing the minimal-spanning tree, a new application of the Kolmogorov-Smirnov test for cluster validity, and a new application of relativistic principles in measures of relationship. Experiments using interactive graphic displays to illustrate these new methods are described, and application of computer programs to meteorological problems is demonstrated. New methodology for research in developing new pattern-recognition methods is illustrated by the development of a closed-loop generation and description language. The same language as that used by the clustering system to describe data is used to generate data. Thus, comparisons of performance of successive research versions of pattern-recognition systems can be easily made, which will facilitate their adaptation from research to application versions.

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

DD FORM 1 NOV 68 1473

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Pattern recognition						
Clustering						
Data analysis						
Statistics						
Artificial intelligence						
Problem solving						
Methodology						
Research						
Computer software						
Man-machine interaction						
Interactive graphics						
Relativity analogy						
Kolmogorov-Smirnov						
Minimal-spanning tree						

*U.S.Government Printing Office: 1973 — 758-429/283

ARL 73-0153

DEVELOPMENT OF NEW PATTERN-RECOGNITION METHODS

D. J. HALL
R. O. DUDA
D. A. HUFFMAN
D. E. WOLF

STANFORD RESEARCH INSTITUTE
MENLO PARK, CALIFORNIA

NOVEMBER 1973

CONTRACT F33615-71-C-1894
PROJECT 7071

Approved for public release; distribution unlimited.

AEROSPACE RESEARCH LABORATORIES
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
WRIGHT—PATTERSON AIR FORCE BASE, OHIO 45433

i-ℓ

FOREWORD

This is the final report for SRI Project 1340, conducted under Contract F33615-71-C-1894 with the Aerospace Research Laboratories at Wright-Patterson Air Force Base. Mr. D. J. Hall, the SRI project leader, is associated with the Computer Science Group of SRI's Information Sciences Laboratory. Dr. D. A. Huffman is a consultant to this group and also teaches Information and Computer Science at the University of California at Santa Cruz. Dr. R. O. Duda is associated with the Artificial Intelligence Center of SRI, and Mr. D. E. Wolf is associated with the Atmospheric Sciences Group of SRI's Radio Systems Division.

The report covers work performed on this project between 1 July 1971 and 19 January 1973.

This contract was technically monitored by Dr. P. R. Krishnaiah of the Aerospace Research Laboratories.

The work was performed at SRI's main offices, 333 Ravenswood Avenue, Menlo Park, California 94025.

ABSTRACT

The problem in pattern recognition is to find a classification or description of the data patterns that matches or suits the data. New methods in pattern recognition are studied in relation to classical approaches using techniques of multivariate statistical analysis. The application of these techniques to specific problems in physical, engineering, behavioral, and other sciences is reviewed. The problems of improved data description and dimensionality reduction are tackled by means of clustering approaches. Several improved clustering methods are developed for general pattern recognition: a new approximate procedure for computing the minimal-spanning tree, a new application of the Kolmogorov-Smirnov test for cluster validity, and a new application of relativistic principles in measures of relationship. Experiments using interactive graphic displays to illustrate these new methods are described, and application of computer programs to meteorological problems is demonstrated. New methodology for research in developing new pattern-recognition methods is illustrated by the development of a closed-loop generation and description language. The same language as that used by the clustering system to describe data is used to generate data. Thus, comparisons of performance of successive research versions of pattern-recognition systems can be easily made, which will facilitate their adaptation from research to application versions.

TABLE OF CONTENTS

Preceding page blank

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

LIST OF TABLES

LIST OF ILLUSTRATIONS

ILLUSTRATIONS (Continued)

ILLUSTRATIONS (Continued)

SECTION I

INTRODUCTION

This introduction will acquaint the reader with the objectives of this study and will show how the work carried out in the study relates to these objectives and to the overall objectives of the Air Force. The pattern-recognition objectives of the surveillance mission of the Air Force were stated in their request for proposal. In this section, we discuss the scope of each item in the work statement related to the overall needs and to each section of the study. We also consider the general objectives of the field of pattern-recognition research, and we explain our motivations in pursuing the particular research directions we have followed. We believe that consideration of these topics in the introduction will contribute to the understanding of the technical sections of this report.

A. The Work Statement and Objectives of This Study

The objectives of this study, as established in the SRI proposal prepared in response to the Air Force request for proposal, are defined in the following tasks:

- Task A--Study the applications of known techniques of multivariate statistical analysis to problems of pattern recognition.
- Task B--Discuss the application of statistical pattern-recognition methodology to specific problems in physical, engineering, behavioral, and other sciences.
- Task C--Develop and study new methods for pattern recognition that provide a better match to the structure of the data. This represents a fundamental attack on the problems of dimensionality in pattern classification.
- Task D--Demonstrate the properties of these new methods by testing them on a specific problem in pattern recognition. This includes the development of computer programs and their use in an experimental verification of the new methods.

Task A is covered in Section II, Task B in Section III, and Tasks C and D in Sections IV to VII. Before making these relationships explicit and describing the extra sections of the report, we expand the discussion of the work statement.

Two key items in the work statement provided by the Air Force are: "The proposed research consists of developing methodology of pattern recognition and investigating the merits of various methods," and "The work to be performed by the contractor involves developing new methodological techniques and discussing their applications," This stress on methodology has led us to attempt a broad view--encompassing many methods--of the discipline of pattern recognition.

B. The Objectives of Pattern-Recognition Research

Research in pattern recognition (hereafter abbreviated p.r.) must be related to the objectives of the practical applications of this research in "real world" situations. These "real world" situations require that reliable p.r. systems be produced that can automatically aid or replace human p.r. functions in a useful way (Task B). Although obvious recognition capabilities, such as reading numbers on bank checks, were already mechanized 10 years ago, these capabilities are extremely inadequate for the subtle human perceptual functions that are employed in real situations. We believe that advances in p.r. will require a deep understanding of human perception psychology. A wide range of modes of human perception exists, ranging from robot-like perception--such as that studied in artificial intelligence--to the vivid perceptions of an intelligent, mature human. It is these latter abilities we must understand, rather than those mechanical attempts at vision that are not yet very successful. Some of our attempts to achieve a deep understanding of human perception psychology may border upon speculation (some parts of Task C), but we

believe that they are worthy of further investigation and development. Our experimental investigations (Task D) are a step toward the deeper understanding of human perception that is the focus of our speculations.

Our purpose in p.r. research for this study is to build upon the best methods of the past, but also to take a fresh approach and to reexamine critically all the known fundamentals of this subject. We have attempted to achieve these diametrically opposed aims; that is, to be completely aware of the latest research results and to remain unbiased by any preconceived opinions contained in past research results reported to date.

The problem of dimensionality in p.r. or classification (Task C) arises when it is not known which measurements may be important for the application--that is, when there is inadequate knowledge about the structure of the data that may arise in an application. Our main attack on this problem of dimensionality is to study and improve clustering methods, since the objective in clustering is to provide a description that is well matched to the structure of the data. Clustering methods (which are relatively new compared to statistics) provide a description of the data being recognized, so that data classification for recognition purposes may be easily made by simply attaching the appropriate recognition labels to the clusters. These labels come, ultimately, from those special applications mentioned in Task B. We have also emphasized a statistical approach to p.r. (see Tasks A and B), since in mechanizing any measurements on data for recognition purposes, this approach is highly promising and attractive.

It is well accepted that computers must be used for p.r. research, and it is becoming increasingly well accepted that man/machine interaction with high performance graphic display is useful for p.r. research (Tasks C and D). We have devised a computer language for clustering, which we

demonstrate in the appropriate section of this report. Of particular significance in this report are the validation of clustering by means of a nonparametric statistic and the development of an approximate method of calculating a "minimal spanning tree," which is useful for purposes of data description.

In the extra section (not specifically mentioned as a task in the work statement), we discuss our computer facilities for p.r. research and the computer programs that have been developed on this project. Finally, we provide an overall summary of our work, drawing conclusions from our labors and making recommendations for future research.

SECTION II

APPLICATIONS OF STATISTICS TO PATTERN RECOGNITION

A. Introduction

Broadly speaking, p.r. is concerned with detecting regularities in a complex and/or noisy environment. When random disturbances are a major factor in a p.r. problem, statistical methods provide an appropriate and powerful tool. In fact, since the publication of a seminal paper by Chow in 1957 [1],[1] the theoretical engineering literature on p.r. has been dominated by statistically oriented papers. The influence of statistics on the design of practical systems has been less pervasive but by no means negligible (Kanal and Chandrasekaran [2]). Thus, statistical theory and practice have had a major influence on research and development efforts in the field of p.r.

This section examines the ways in which various topics in statistics relate to p.r. problems, with particular emphasis on statistical approaches to clustering problems. Our purpose is to provide an overview, not a comprehensive literature survey or a tutorial exposition. Many excellent literature surveys[2] and several fine textbooks are available, and there is no need for another long compilation of references. Rather, our intention is to place this work in perspective, to mention some of the key contributions, to point out some of the limitations, and to suggest possibly fruitful areas for future work.

---

[1] References are listed at the end of the report.

[2] Among the more than 40 surveys of which we are aware, the general survey by Nagy [3] is particularly valuable. The survey of classification algorithms by Ho and Agrawala [4], the statistically oriented survey of clustering by Bolshev [5], and the computer-oriented survey of clustering by Ball [6] are also highly recommended.

B. Statistical Decision Theory

Many p.r. problems are classification problems. An object or event is sensed by an appropriate transducer; a vector x of characterizing features (properties, measurements, variables, characteristics, items, and attributes) is computed; and on the basis of this information, the object or event is assigned to one of a predetermined number c of classes. Problems in character recognition, target detection, blood cell classification, spoken word recognition, and radar signal detection, among others, all fit this description. As was pointed out by Chow [1], such problems of decision making under uncertainty fall in the domain of statistical decision theory. In the Bayesian formulation, if the class-conditional probability density functions $p_i(x)$ and the class a priori probabilities P_i, $i = 1,\ldots,c$, are known, then the (Bayes) optimal classification rule for minimum probability of error is merely to choose that class for which $p_i(x)P_i$ is maximum. Other formulations of the decision problem are sometimes more appropriate, but the majority of pattern classification problems can be posed in Bayesian terms.

A virtually universal characteristic of interesting pattern classification problems is that the dimensionality d of the feature vector is quite large. In character recognition applications, for example, measurements of 50 or 100 different features for each character are typical. Thus, p.r. problems are fundamentally problems in multivariate statistics. The great difficulties encountered in working with general multivariate densities have tempted many workers to assume statistical independence, thereby reducing a multivariate density to a product of univariate densities. However, independence assumptions are rarely justified in practice and must always be considered suspect.

Another common assumption is that the densities are multivariate normal (Marill and Green [7]). This assumption has the virtue of allowing for correlated features while maintaining considerable analytical

simplicity. The optimal decision rule merely requires evaluating c linear (or, at worst, quadratic) discriminant functions that are simply related to the parameters of the densities, the mean vectors μ_i, and the covariance matrices Σ_i. In the special case of equal a priori probabilities and equal covariance matrices, the rule reduces to assigning x to the class for which the squared Mahalanobis distance $(x-\mu_i)^t \Sigma^{-1}(x-\mu_i)$ is minimum. This is equivalent to choosing the class for which the linear discriminant function $g_i(x) = x^t\Sigma^{-1}\mu_i - \frac{1}{2}\mu_i^t \Sigma^{-1}\mu_i$ is maximum.

The multivariate normal assumption is reasonable, at least as a first approximation, if the values of each feature for objects in a given class can be described as noisy versions of some ideal or prototype value. Stated geometrically, a random sample of points drawn from one of the c populations must fall into a single, ellipsoidally shaped cloud centered about the mean vector. Some designers take the viewpoint that part of the task of designing a feature set is to transform or normalize the features so that they satisfy this requirement. However, this is not always easily achieved, and in some applications (particularly those arising in the biological and sociological sciences), one has little or no control over the nature of the features. Much of the recent interest in clustering stems from a desire to represent the actual density as a mixture of normal densities, i.e., to find normal subclasses that can be separately treated by the relatively simple, multivariate normal model. Section II-F discusses this topic further.

In many applications, the features are binary valued, and the multivariate normal model provides a crude approximation at best. Simple, exact solutions are available for the independent (or "multivariate Bernoulli") case, which again leads to linear discriminant functions (Minsky [8]). The Bahadur-Lazarsfeld expansion [9] provides a sequence of successively more accurate approximations that take various orders of dependence into account. However, when the dimensionality is high, it is difficult

to account for all possible second-order correlations, to say nothing of higher order correlations. A promising approach to treating this problem stems from the notion of tree dependence introduced by Chow in 1966 [10] and explored by Chow and Liu in 1968 [11]. This idea for finding the basic relations between variables can also be applied to the multivariate normal case, and it deserves greater attention from the designers of p.r. systems.

C. Parameter Estimation

The most difficult problems encountered in applying multivariate statistics to p.r. involve the class-conditional densities $p_i(x)$. Even when it is possible to assume that these densities have simple parametric forms, the estimation of parameter values is not trivial. Consider, for example, the multivariate normal case where $p(x) \sim N(\mu,\Sigma)$. Since neither the mean vector μ nor the covariance matrix Σ are usually known, it is customary to estimate these values from a so-called design sample or training sample $\mathcal{X}$ of n independently drawn instances $x_1,\ldots,x_n$. The obvious approach is to use the maximum likelihood estimates

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

and

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n} (x_k - \hat{\mu})(x_k - \hat{\mu})^t$$

as if they were the true parameter values in designing the classifier.

There are both theoretical and practical objections to the approach. One theoretical objection is that consistency can only be demonstrated in the large-sample case. Although this procedure is a reasonable approach in the finite-sample case, it has no demonstrably optimal properties. In a classic paper, Abramson and Braverman [12] showed how the

sample could be used optimally in a Bayesian sense for the case where only the mean vector is unknown. The basic idea involves the recursive computation

$$p(\mu|x_1,\ldots,x_n) = \frac{p(x_n|\mu)\,p(\mu|x_1,\ldots,x_{n-1})}{\int p(x_n|\mu)\,p(\mu|x_1,\ldots,x_{n-1})\,d\mu}$$

which starts with an (assumed known) a priori density $p(\mu)$ for the unknown mean vector. The convergence of the sequence of functions $p(\mu)$, $p(\mu|x_1)$, $p(\mu|x_1,x_2),\ldots$, is called Bayesian learning, and the basic approach has been extended to cover parametric learning in general (see, for example, Geisser [13]).

One practical objection is that a d-dimensional covariance matrix contains d(d+1)/2 independent elements, which leads to considerable storage requirements when d is large. Even more important, it imposes a great demand for data to obtain a useful estimate. If the sample size does not exceed the dimensionality, the maximum likelihood estimate $\hat{\Sigma}$ is singular and useless for classifier design. Even if $\hat{\Sigma}$ is not singular, chance correlations due to small sample size can degrade the performance of the classifier below that provided by a simpler classifier that uses only a subset of the d features. Stated another way, the performance of a classifier designed for a sample of fixed size will not improve indefinitely as the number of features is increased; it will reach a peak and then decline (Kanal and Chandrasekaran [14]). This has been one of the major reasons for interest in methods of dimensionality reduction, such as those discussed by Meisel [15]. Although the basic mechanisms behind this phenomenon are at least partially understood, no useful solutions are known, and there is a real need for further understanding of these topics.

D. Nonparametric Methods

Because classical models, such as the multivariate normal model, frequently fail to provide adequate approximations to the unknown densities, various investigations of nonparametric methods have been pursued. Some of these are basically theoretical and are more concerned with what is possible in principle than in practice (see Fix and Hodges [16]). Thus, while very general nonparametric methods for estimating probability densities are of fundamental theoretical importance, they are rarely used in practice because of their enormous requirements for data storage and computation time. To alleviate these problems, various approximate procedures have been advanced. The well-known heuristic histogram method of Sebestyen and Edie [17] is one of the few procedures of this type that has actually been used. It is interesting to note in passing that this procedure was not originally developed as an approximation to an exact estimation procedure, but rather as an extension of a mode-seeking clustering procedure. The goal of finding the modes of an unknown density is common to many nonparametric estimation and clustering procedures.

Nearest-neighbor classification methods provide alternative nonparametric procedures for using the samples for pattern classification. The simplest of these is the single-nearest-neighbor method, in which an unknown vector x is assigned to the class of the nearest instance x_i. Cover and Hart [18] showed that the large-sample error rate for this procedure could not exceed twice the Bayes error rate, which may be quite satisfactory performance when the Bayes rate is small. Unfortunately, no similar theoretical properties are known for the finite-sample case, and once again the data storage and computation time requirements are severe. A natural approach is to look for ways to discard instances that are no more or less redundant, thereby reducing the sample size to manageable

proportions. Some attempts along this line have been made (see Wilson [19]), but the practical value of such methods has yet to be established.

An even less explored area is the use of nonparametric tests of goodness of fit for p.r. applications. Considering the controversy that has often attended the employment of normality, it may seem surprising that so few attempts have been made to test the validity of these assumptions. The main reason that more has not been done is that the classical measures of goodness of fit (as described, for example, in Gibbons [20]) are ill suited for multivariate distributions. One can, of course, test the univariate marginal distributions, but meaningful multivariate tests again impose severe requirements for data and computation, and it is often simpler to let classification performance justify the assumptions.

E. Discriminant Analysis

While the use of linear discriminant functions for classification can be traced to a classic paper by Fisher [21], much of the enthusiasm for linear discriminants for p.r. applications came from other sources, including neural-net brain models, switching theory, and the mathematics of linear inequalities. The Perceptron work of Rosenblatt [22] stimulated interest in the adaptive or learning aspects of linear discriminants, with error-free performance being the goal, and the influential paper by Highleyman [23] emphasized the nonparametric aspects and the advantages of computational simplicity.

Indeed, simplicity is probably the chief virtue of this approach. It leads naturally to special purpose hardware implementation, and several such systems have been built (see Brain et al. [24]). It also leads to tractable analytical problems; consequently, the literature on this topic is vast. Duda and Hart [25] treat the major topics (the Perceptron and relaxation procedures, minimum-squared-error methods, stochastic

approximation, the method of potential functions, and linear programming techniques) and provide many references to the literature. Although there is considerable theoretical interest in the different properties of these various methods, in practice these methods usually provide essentially the same performance, with the limitations due more to the inherent limitations of hyperplane decision surfaces than to the ways in which the coefficients are computed.

In theory, increased separating power can be obtained by using quadratic or polynomial discriminant functions. However, the number of undetermined coefficients in a multivariate polynomial of degree m is of the order m^d. This imposes comparable requirements for sample size to determine these coefficients and for computation to evaluate the discriminant functions. Thus, it is rare to find even quadratic discriminant functions in practice. An attractive alternative is the use of piecewise linear discriminant functions. Piecewise linear decision surfaces can provide good approximations to more complicated decision surfaces with far fewer variable coefficients. However, despite some interesting attempts, there are no solutions comparable to the solutions for linear discriminants. At present, the best procedure for obtaining piecewise linear discriminants is to use a clustering procedure to split the data into subclasses and to obtain linear discriminants for each subclass.

F. Clustering

1. General Remarks

As we have observed, a major reason for interest in clustering for p.r. has been the desire to find a compromise between parametric techniques using the multivariate normal model, which often fails to provide an acceptable description of the data, and general nonparametric techniques, which may impose impossible demands for storage and computation

time. A natural approach is to try to represent the population as a mixture of normal subpopulations. Although the analytical simplicity of the normal model is lost, the mixture model is much more flexible and is still computationally acceptable. However, this is far from the only approach to clustering. A great variety of techniques, based on different orientations and user goals, have been proposed and investigated. Users concerned with computational efficiency have favored minimum-squared-error partitions, users concerned with constructing taxonomies have favored hierarchical procedures, and users concerned with nonmetric data have favored graph-theoretic methods.

This section considers all of these approaches. Although the treatment here is somewhat more detailed than the preceding discussion, it is still an overview rather than a survey. Earlier reference was made to the statistically oriented survey by Bolshev [5] and the computer-oriented survey by Ball [6]. Other useful general surveys are given by Nagy [3] (1968--43 references), Ball [26] (1970--135 references), Ling [27] (1971--140 references), and Dorofeyuk [28] (1971--204 references, including many Russian papers). More specialized surveys by Spragins [29] and Cooper [30,31] cover topics related to communication theory, and those by Williams and Dale [32], Sneath [33], and Wishart [34] cover topics related to numerical taxonomy.

Tutorial presentations of clustering for p.r. applications are given by Meisel [15] and Duda and Hart [25]. The books by Sokal and Sneath [35] and Jardine and Sibson [36] treat techniques for numerical taxonomy, and the book by Tryon and Bailey [37] primarily treats the clustering of features. Philosophical foundations of clustering are examined by Watanabe [38]. Finally, several papers have included interesting critiques of clustering, including those by Fleiss and Zubin [39], Wishart [34], Jardine and Sibson [36], and Ling [27]. Dissatisfaction

with present methods will no doubt continue to provide strong motivation for further developments in clustering.

2. Mixture Decomposition

In the normal mixture model, it is assumed that the sample $\mathcal{X} = \{x_1, \ldots, x_n\}$ is drawn from the mixture density

$$p(x) = \sum_{i=1}^{c} p_i(x) P_i$$

where the component densities $p_i(x)$ are normal with mean vectors μ_i and covariance matrices Σ_i. The problem is to use the sample to determine these parameters, the so-called mixing parameters P_i, and possibly the number c of components. The simple, univariate, two component case was investigated by Karl Pearson [40] as early as 1894. Pearson used moment estimators, a robust approach that has been extended by several researchers (see Cooper [31]). Unfortunately, this approach is very complicated when there are more than two components in the mixture.

Other standard approaches for estimating the unknown parameters, such as minimum χ^2, Bayesian, and maximum likelihood methods, also encounter severe problems. The maximum likelihood approach actually leads to degenerate, singular solutions when the covariance matrices are not restricted in some way (Day [41]). Despite this defect, it provides the most effective, exact procedure for decomposing multicomponent, multivariate normal mixtures. A clear derivation of the maximum likelihood equations is given by Wolfe [42-44], whose NORMIX program can accommodate as many as 20 components, 10 features, and 1000 individuals (Wolfe [45]).

Many heuristic procedures for clustering can be thought of as approximate procedures for mixture decomposition. When the components of the mixture are well separated--i.e., when the squared Mahalanobis

distances $(\mu_i - \mu_j)^t \Sigma^{-1}(\mu_i - \mu_j)$ are large--the mean vectors locate the modes of the mixture density. Thus, many mode-seeking procedures (Sebestyen [46], Ball and Hall [47]) can be viewed as ways to estimate the component means. Such procedures can be used to partition the sample on a minimum-distance basis into subclasses corresponding to the various components, and then the parameters for each component can be estimated separately. Of course, such procedures are biased and generally do not have optimal large-sample properties. In practice, however, most arguments about the relative merits of "exact" and "approximate" procedures are rather academic. More important problems remain largely unsolved, such as determining the number of components, the validity of a cluster description, cr the appropriateness of the normal mixture model. Some of these issues are discussed in Section IV-B. The validity of cluster descriptions is also treated by means of the closed-loop recognition methodology and experiments covered in Section VI-A-3.

3. Minimum-Squared-Error Partitions

When not enough is known about the problem to specify the forms of the distributions of subpopulations, an alternative approach is to specify a criterion function that measures how well a given partition of the sample divides it into coherent groups. Then the problem is to find the partition that optimizes the criterion function. The most widely used (and hence most frequently criticized) criterion is the sum of squared errors. With this criterion, the individuals in the ith cluster $\mathcal{X}_i$ are represented by the sample mean m_i for that subset, and the criterion function is merely

$$J = \sum_{i=1}^{c} \sum_{x \in \mathcal{X}_i} (x - m_i)^2$$

Although this criterion function has a simple form and some convenient analytical properties, its minimization is far from trivial. The only procedures that are guaranteed to yield the optimal partition are exhaustive and impossibly time consuming (see Fortier and Solomon [48]). Thus, approximate procedures, such as the ISODATA procedure of Ball and Hall [47] must be used. Wishart [34] provides a good, brief survey of 13 of the better known methods.

The minimum-squared-error approach has been generalized in various ways, most notably with the introduction of invariant criteria by Friedman and Rubin [49]. The approach has also been criticized, primarily on the ground that the resulting, implicitly defined clusters may bear little relationship to the true structure of the data. Nevertheless, this approach remains one of the most popular ones for p.r. applications.

4. Hierarchical Clustering

Clustering in the biological sciences is dominated by hierarchical groupings, where individuals are grouped to form species, species are grouped to form genera, genera are grouped to form families, and so on. Works on numerical taxonomy, such as the well-known book by Sokal and Sneath [35], devote virtually exclusive attention to this kind of clustering. One of the best known procedures for hierarchical clustering is the single-linkage algorithm, which can be described as follows. Initially, the sample $\mathcal{X} = \{x_1, \ldots, x_n\}$ is partitioned into n clusters, the ith cluster $\mathcal{X}_i$ containing the ith individual x_i. Next, the closest pair of individuals is found, and the number of clusters is reduced by one by merging those two clusters. In general, such mergers will cause the clusters to contain more than one individual. In the single-linkage algorithm, the distance between two clusters $\mathcal{X}_i$ and $\mathcal{X}_j$ is defined as the distance between the nearest individuals, one in $\mathcal{X}_i$ and the other in $\mathcal{X}_j$.

Then, the general procedure is to reduce successively the number of clusters by repeatedly merging the nearest pair of clusters.

There are numerous variations of this procedure, but all of them possess the following characteristics: (1) The clusterings are hierarchical, and (2) the computation time grows as n^2. The hierarchical structure implies a tree representation of the data, which may or may not accurately describe the actual structure present. The very interesting paper by Hartigan [50] is one of the few that addresses this important question about cluster validity. The computational problem is serious only when the sample size n is large (say, greater than 500). The work reported in Section IV-C is relevant to solving this problem.

5. Graph-Theoretic Methods

It can be shown that the single-linkage algorithm is actually an algorithm for generating a minimal spanning tree--a tree whose edges join the individuals and whose sum of edge lengths is as small as possible (Gower and Ross [51]). Zahn [52] presents several properties of minimal spanning trees that make their use in clustering appear quite attractive, particularly for situations in which the normal mixture model is clearly inappropriate.

Graph-theoretic concepts, such as the concept of maximal, complete subgraphs, provide new ways of viewing clustering problems and have been exploited in such areas as information retrieval, where structural complexity rather than randomness seems to present the clustering problem. In general, combining statistical and graph-theoretic methodologies has appeared so difficult that little work in this area has been attempted. A noteworthy exception is the thesis by Ling [27], in which a graph-theoretic procedure is proposed and its distributional properties are studied. More work of this sort may bring us closer to a clear understanding of the process of using clustering to find and describe the structure in a set of data.

SECTION III

APPLICATIONS OF PATTERN-RECOGNITION TECHNIQUES

A. Introduction

Statistical techniques for p.r. have been applied to a great many problems of commercial, social, military, and scientific interest. Some of these applications have been major engineering projects, where the classification problems have been only one aspect of larger systems considerations. Others have been small studies, often aimed as much at illustrating a particular technique as at solving a specific problem. One of the unfortunate consequences of all of this work has been the accumulation of a vast collection of reports and papers for a relatively few effective and working p.r. systems.

One of the most comprehensive surveys of p.r. applications is that of Stevens [53]. In addition to surveying the regular literature, her valuable report contains brief descriptions of commercial systems--such as the Kartrak system widely used to identify and monitor railroad cars--that are hard to find in any other source. An exceptionally clear description of the state of the art in eight specific application areas is given by Nagy [3]. Although some advances have occurred in each of these areas since Nagy's paper was written (in 1968), the descriptions remain sound, and they convey well the nature of the problems in each area. A survey of techniques for recognizing hand-printed characters was conducted by Chodrow, Bivona, and Walsh [54] and a survey of techniques for recognizing cursive script and speech was performed by Lindgren [55]. Beyond these surveys, there is little information available to aid in the evaluation of the true state of the art in p.r.

Preceding page blank

Our purpose in this section is to provide an overview of the major application areas to which statistical p.r. techniques have been applied. Although we cite a number of references, they represent only a small fraction of the published literature; for more detailed information, the surveys mentioned in the preceding paragraph should be consulted. The overview is divided into three parts: first, pictorial applications; second, waveform applications; and third, applications to more abstract kinds of data, such as questionnaires. Although not complete, these areas cover the major applications of statistical p.r. methodology.

B. Pictorial Data Analysis

By far the most important commercial application area is that of optical character recognition. An excellent capsule survey of the various kinds of available document readers, journal tape readers, and page readers is given by Andersson [56]. The most common of these are single-font readers that rely on statistically primitive template matching procedures, augmented by various ingenious engineering techniques. The more complicated multifont readers have benefited from more sophisticated classification methods, but these rarely play a central role in the overall design (Andrews et al., [57]). However, several very well-performed research studies at the IBM Watson Research Center have exploited many statistical techniques, including some very interesting applications of clustering (see Casey and Nagy [58]).

More challenging problems are presented by the recognition of hand-printing and cursive script. Although several commercial readers are capable of reading constrained hand-printing, this remains a difficult task, and the problem of reading cursive script is virtually unsolved. However, both of these problem areas are relatively well documented, as can be appreciated from the survey by Chodrow, Bivona, and Walsh [54], the state-of-the-art paper by Munson [59], and the survey by Lindgren [55].

Several technically successful attempts have been made to recognize hand-printed characters drawn on-line with an input device, such as a light pen or graphics tablet (see Groner [60]). These techniques promise to be particularly valuable as a part of systems for analyzing line-drawing data, but such systems have been held back by economic considerations and less than perfect performance. A good overview of the problems encountered in processing line-drawing data is given by Freeman [61].

Considerable effort has been devoted to automatic methods for personnel identification, although much of this work has been kept secret and unreported. A partial exception is fingerprint identification (or fingerprint verification), which has received the attention of several laboratories (see Wegstein et al. [62]). A major effort to automate this process is currently being carried out at the Calspan Corporation. Some related but even more difficult problems are those of face verification and voice-print verification. Although some progress has been made in these areas, these problems are far from completely solved.

Another area that has received considerable attention is the automatic screening and detection of targets in aerial photographs. The design of such systems as Litton's Automatic Target Recognition Device (ATRD) exploited a number of statistical procedures (Swoboda and Gerdes [63]). However, the tyranny of numbers (speed and accuracy requirements) severely constrains the approaches that can be used. A lucid, convincing discussion of how systems requirements dictate the techniques that are feasible is given by Harley et al. [64]. Hawkins [65] presents supporting data on the hardware requirements for practical, automatic image processing.

Work in biomedical image processing has been carried out at a large number of laboratories, but many of these efforts have been relatively small, pilot study projects. Typical problems studied have included leukocyte classification (Mendelsohn and Prewitt [66]), chromosome analysis

(Ledley and Ruddle [67]), and the analysis of radiographic images, such as chest X-rays (Dwyer et al. [68]). Some of this work, such as the studies of leukocyte classification, has made extensive use of statistical techniques. Those image-processing problems that had description as the primary goal have made little or no use of statistics. This is also true of a number of other interesting fields in scene analysis, including the analysis of bubble chamber photographs and the development of visual systems for robots. As the problems of complexity in these fields are solved, however, greater attention may well be paid to the problems of randomness that are now solved by ad hoc techniques.

C. Waveform Data Analysis

The communication theory use of statistical techniques to detect signals in noise antedated the development of p.r. as an established discipline. However, since signal detection problems can be viewed as a special class of p.r. problems, there is an obvious overlap between communication theory and p.r. In addition to standard communications problems, decision theory has been widely applied to the detection of radar and sonar echoes. Of the many books that have been written on the theory of signal detection, we can cite the volume by Middleton [69], the text by Helstrom [70], the monograph by Selin [71], and the series of books by Van Trees [72-74].

Among the applications that fall strictly within the province of p.r., the most work has been done in the area of speaker and speech recognition. Work on identifying or verifying the identity of a speaker from a sample of his speech is reviewed in an excellent survey by Hecker [75]. The most fashionable procedure in current use is the voiceprint method, but the reliability of the method is still disputed, and much work remains to be done.

Work on speech recognition is surveyed by Lindgren [55], Hyde [76], Young and Hecker [77], Hill [78], and Otten [79]. Most of this work has been restricted to the recognition of relatively small vocabularies of isolated words, without the benefit of linguistic or semantic constraints. Typical approaches include the format-based system of Hemdal and Hughes [80], the spectral analysis approach of Martin et al. [81], and the only partially implemented analysis-by-synthesis approach of Halle and Stevens [82]. The system of Vicens and Reddy [83] was one of the first serious attempts to deal with connected speech. The study report by Newell et al. [84], stresses the importance of incorporating syntactic and linguistic constraints for the recognition of connected speech, and it outlines the major components of a speech understanding system. Research on such a system is currently being performed at SRI.

Other work on waveform analysis has been scattered over a variety of specialized fields. In the biomedical area, computerized methods have been devised for analyzing both electrocardiograms and electroencephalograms. Attempts have been made to distinguish automatically between nuclear explosions and earthquakes from seismograms. Various proprietary studies have been made of seismogram analysis for oil exploration. Techniques have been developed for distinguishing between normal and various classes of faulty engines on the basis of records of engine noise. Studies have been made of the identification of airplanes and other military vehicles from data from sonic sensors. The variety of these problems make it difficult to summarize them, other than to say that statistical decision procedures have been applied to virtually every problem.

D. Mathematical Data Analysis

Pattern recognition problems arise in many scientific fields, including chemistry, geology, mineralogy, meteorology, paleontology, biology, medicine, demography, sociology, library science, and economics. Many of these problems have been studied for years and are just as well described less pretentiously as statistical classification problems (Rao [85]).

Probably the chief contribution of p.r. research to these traditional fields of learning has come through developments in clustering. In biology, this has primarily meant the special techniques of numerical taxonomy (Sokal and Sneath [35]). In psychology, the primary emphasis has been on multidimensional scaling (Shepard and Carroll [86]). Other applications of clustering have been quite diverse, ranging from determining subclasses of kidney diseases to designing feedback control algorithms for industrial plants. The 83 references given by Dorofeyuk [28] contain illustrations of most of the published applications of clustering techniques to scientific and technical problems.

SECTION IV

DEVELOPMENT OF NEW METHODS

A. Introduction

The research described in this report has addressed several fundamental problems in p.r. In one way or another, all of them involve the development of new methods and techniques for clustering multivariate data.

One basic question is how to determine the number of clusters that exist in a given set of data. A related question is measurement of how well a given cluster description describes a given set of data. We address both of these questions in Section IV-B, where the use of a multivariate extension of the Kolmogorov-Smirnov test is proposed and studied.

Another fundamental problem is computational cost. Indeed, the search for computationally feasible alternatives to theoretically optimal, exhaustive methods has dominated much of the work directed at optimizing criterion functions. To date, little attention has been given to the computational cost of the attractive graph-theoretic approaches. Section IV-C discusses the computational requirements of standard methods for finding minimal spanning trees, and it presents and evaluates a new, much more efficient, approximate procedure.

B. Cluster Validity

1. Background

For many years, techniques for summarizing and describing data formed a major topic in statistics. Descriptive statistics is now usually considered as a means to the end goal of decision making, rather than an end in itself. However, to use decision-theoretic methods, one must

hypothesize or estimate underlying distributions. The failure of such familiar models as the multivariate normal model to describe adequately the data for many p.r. applications stimulated much of the interest in clustering, which promised to provide better models for multivariate data. Thus, the study of clustering reflects a rebirth of interest in descriptive statistics.

When a representation in terms of clusters is proposed as a way to describe a data set, it is natural to ask in what way the resulting description is better than simpler alternatives, and how well it describes the data set. Unfortunately, even such a simple-sounding question as, How many clusters are actually present? cannot be answered without making rather strong assumptions about the nature of the data. In general, the best one can do is hypothesize something about the structure of the data, and then perform a test to accept or reject the hypothesis.

The probable reason that so few attempts have been made to devise tests for the validity of clusters is that tests that are meaningful, computationally feasible, and analyzable are difficult to find. Most of the tests that have been suggested possess no known distributional properties. For clustering procedures based on minimizing a criterion function, it is natural to observe how the criterion function decreases as the number of clusters increases, and to look for a "knee" in the curve that might reveal the number of clusters present (Ball [26]). In using a criterion function J to evaluate a clustering, one would like to know the sampling distribution of J under the null hypothesis that no cluster structure is present. Hall, Tepping, and Ball [87] provided a partial answer to this question by deriving the large-sample expected value of J using a uniform distribution for the null hypothesis.

Bonner [88] proposed testing the validity of a cluster $_i$ of size n_i extracted from a larger sample $\mathcal{Y}$ of size n by comparing the mean m_i for $\mathcal{Y}_i$ to the mean m for $\mathcal{Y}$. His test statistic has a χ^2 distribution under the null hypothesis that the n_i objects in $\mathcal{Y}_i$ were drawn randomly from $\mathcal{Y}$. Of course, almost any clustering procedure will produce a more homogeneous grouping than will a random selection. Thus, Bonner suggested an interesting (but not necessarily tight) worst-case bound to obtain a critical value for accepting or rejecting the null hypothesis. A quite different approach was advocated by Hartigan [50] who used a weighted sum-of-squared-error criterion to measure how well a given hierarchical clustering represented the interpoint distances (or dissimilarities) of a set of data. Hartigan made some speculations concerning the sampling distribution of this statistic but observed that there was little hope for obtaining exact answers for other than the simplest cases.

Among the most rigorous attempts to obtain tests with known distributional properties were those of Ling [27], Bargmann and Garney [89], and Wolfe [44]. Ling proposed a graph-theoretic clustering procedure that produces a variety of alternative clusterings of a data set. Some of these are subsets of others, with all of them being a subset of the entire data set. Ling defines an "isolation index" based on the depth of nesting of these subsets that is a plausible measure of the "reality" of each cluster. Using analysis based on the theory of random graphs and using Monte Carlo simulations, he obtained a variety of results on the sampling distribution of the isolation index.

Bargmann and Garney use the more specialized normal mixture model. By projecting the data from a hypothesized single cluster onto the surface of a surrounding hypersphere, they obtain a uniform distribution of data points. Any clusters found here are called virtual clusters,

since they may not correspond to clusters in the original space. Bargmann and Garney develop a simple test of the null hypothesis that the sample comes from a normal population, and they provide tables for their angular test statistic [90].

Wolfe's approach, which is also based on the normal mixture model, is considerably simpler. He observes that the maximum value of the likelihood function L(c) will increase as the number c of clusters is increased to c', but perhaps not enough to be statistically significant. Appealing to a general asymptotic property of likelihood functions, he asserts that the distribution of $\chi^2 = -2 \log(L(c)/L(c'))$ is χ^2 with m degrees of freedom, where m is the difference in the number of parameters estimated for each case. Thus, Wolfe keeps increasing the number of clusters until χ^2 is less then, say a 1-percent or 5-percent critical value.

The approach proposed in this section is to use a multivariate version of the Kolmogorov-Smirnov (KS) test to decide whether a given data set is adequately described as a sample from a normal population. If the null hypothesis is accepted, no further clustering is considered justified. If it is rejected, any of a variety of clustering procedures can be used to cluster the data further, and the KS test can be applied separately to each resulting cluster.

This approach is similar to Wolfe's in two ways: It is based on an assumed normal mixture model, and it is directed at sequentially finding the smallest number of clusters adequate to describe the data. It differs in that it provides a direct measure of goodness of fit, and it provides separate measures for each cluster, rather than a combined measure for all of the clusters. Thus, it is less likely to accept a cluster description that fits a subset of the data poorly merely because that subset is small.

2. The Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test is a well-known nonparametric procedure for measuring goodness of fit. The properties of this and related tests are well described in the survey by Darling [91] and the text by Gibbons [20]. Thus, the following summary is given primarily to establish notation.

Let $\mathcal{X} = \{x_1,\ldots,x_n\}$ be a sample of size n from a univariate population, the observations x_i being independent random variables sharing a known continuous distribution function F(x). Let

$$u(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

and define the empirical distribution function $F_n(x)$ by

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} u(x - x_i)$$

Since, by the Glivenko-Cantelli theorem, $F_n(x)$ converges uniformly to F(x) as n goes to infinity, the Kolmogorov-Smirnov one-sample statistic

$$D_n = \sup_x \left| F_n(x) - F(x) \right|$$

converges to zero (in probability) as n goes to infinity. Moreover, if F is continuous, the supremum or maximum is achieved at one of the sample points. A fundamental property of this statistic for finite n is that its distribution is independent of F(x), provided only that F is continuous.

Since the asymptotic distribution of $F_n(x)$ is $N\left[F(x), nF(x)\,[1 - F(x)]\right]$, one might expect $\sqrt{n}\,D_n$ to have a well-behaved limiting distribution. Indeed, Kolmogorov showed that for any continuous F(x),

$$\lim_{n \to \infty} P(\sqrt{n}\,D_n > z) = 2 \sum_{i=1}^{\infty} (-1)^{i-1}\, e^{-2i^2 z^2}$$

This asymptotic distribution is shown graphically in Figure 1, and is a good approximation to $P(\sqrt{n}\,D_n > z)$ for all n greater than about 25. Exact values for small values of n are tabulated by Birnbaum [92].

The use of the KS statistic to test the null hypothesis that F(x) is the distribution function for the sample $\mathcal{Y}$ is straightforward. One selects an acceptable value for the Type I probability of error α and determines a corresponding critical value z_α such that $(P\sqrt{n}\,D_n > z_\alpha) = \alpha$. Then the null hypothesis is rejected at the α level if the observed KS statistic satifies $\sqrt{n}\,D_n > z_\alpha$. For large n, $z_{0.05} = 1.35$ and $z_{0.01} = 1.63$.

3. Problems in Using the KS Test

In extending the KS test to test the validity of a cluster, three problems arise: The distribution function F(x) is not known exactly, the observation x is a d-dimensional vector rather than a scalar, and the test is no longer distribution-free. All of these facts cause serious problems, some theoretical and some computational.

Consider first the fact that F(x) is not known exactly. If we assume that the distribution is multivariate normal with unknown mean μ and unknown covariance matrix $\underline{\Sigma}$, we can use the sample to obtain, say, maximum likelihood estimates for μ and $\underline{\Sigma}$:

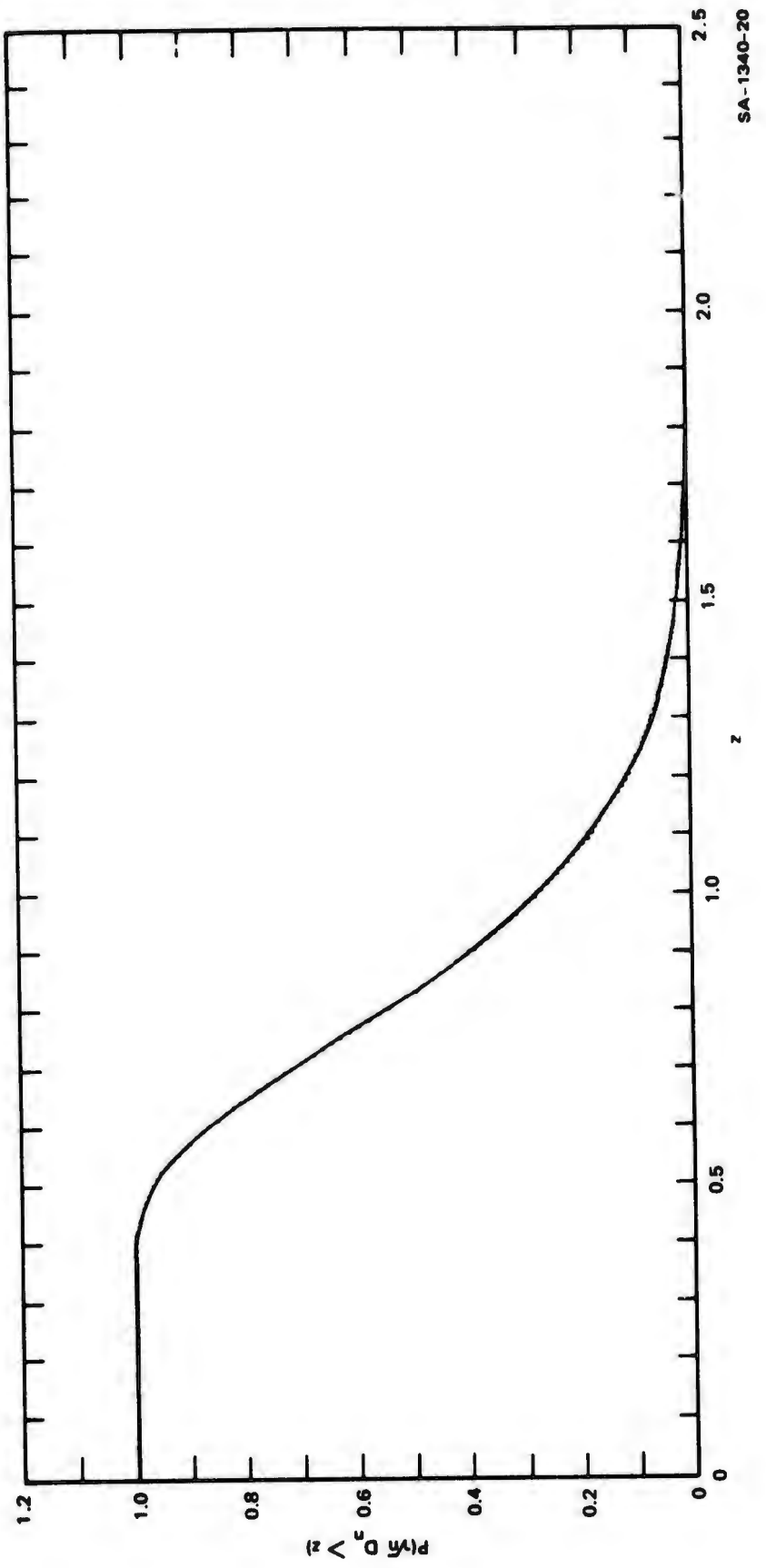


Figure 1. Asymptotic Distribution for $\sqrt{n}\, D_n$

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\hat{\underline{\Sigma}} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})(x_i - \hat{\mu})^t$$

Unfortunately, the estimate $\hat{F}(x)$ obtained from using these estimates as if they were the true parameter values results in significantly smaller values of $\hat{D}_n = \sup \left| F_n(x) - \hat{F}(x) \right|$, invalidating the use of the sampling distribution for D_n to determine critical values of $\hat{D}_n$. Moreover, the discrepancy does not become negligible in the large-sample case, basically because D_n and the error in estimating μ and $\underline{\Sigma}$ (and hence F) both approach zero as $1/\sqrt{n}$. Since analysis of this problem is very difficult, one must resort to Monte Carlo simulations to obtain the distribution for $\hat{D}_n$ when parameters of F(x) are estimated from the sample. Lilliefors [93] gives results of such simulations for the univariate normal case, and we give multivariate normal results in Section IV-B-5.

Consider next the new problems that arise in the multivariate case. Let x be the d-dimensional vector $x = (x^1,\ldots,x^d)^t$, and define

$$u(x) = \begin{cases} 1 & x^i \geq 0,\ i = 1,\ldots d \\ 0 & \text{otherwise} \end{cases}$$

Then the formal definition of the empirical distribution function is the same as before:

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} u(x - x_i)$$

Consider, however, the problem of computing the KS statistic

$$D_n = \sup_x \left| F_n(x) - F(x) \right|$$

In the univariate case, a well-known consequence of the assumed continuity of $F(x)$ is that the supremum is achieved at one of the n observation points. Thus, to compute D_n, one need only evaluate $\left| F_n(x) - F(x) \right|$ at each of the n points $x = x_i$, and, since the computation of $F_n(x)$ grows linearly with n, the overall computation grows only quadratically with n.[3]

Unfortunately, in the multivariate case it is no longer true that the supremum is always achieved at one of the n observation points. Consider, for example, the empirical distribution function $F_n(x)$ shown in Figure 2. In addition to being discontinuous at the five data points, $F_n(x)$ is discontinuous along all of the straight line segments shown. Of course, if $F(x)$ is continuous in both x^1 and x^2, one can restrict the search for the supremum of D_n to those points where $F_n(x)$ is discontinuous in both x^1 and x^2. However, these points include the circled intersection points in addition to the data points. Moreover, the supremum is actually quite likely to be achieved at one of these intersection points rather than at a data point. There are three reasons for this. First, in d dimensions, $F_n(x)$ jumps by 1/n at a data point but by as much as d/n at an intersection point. Second, when tight clusters are present, large changes in $F_n(x)$ occur around the intersection points for the cluster centers (see Figure 3). Third, in high dimensional spaces, there are

---

[3] In fact, if one sorts the observations first (an n log n operation), $F(x_i) = i/n$, and the computation is limited by the n log n sorting time.

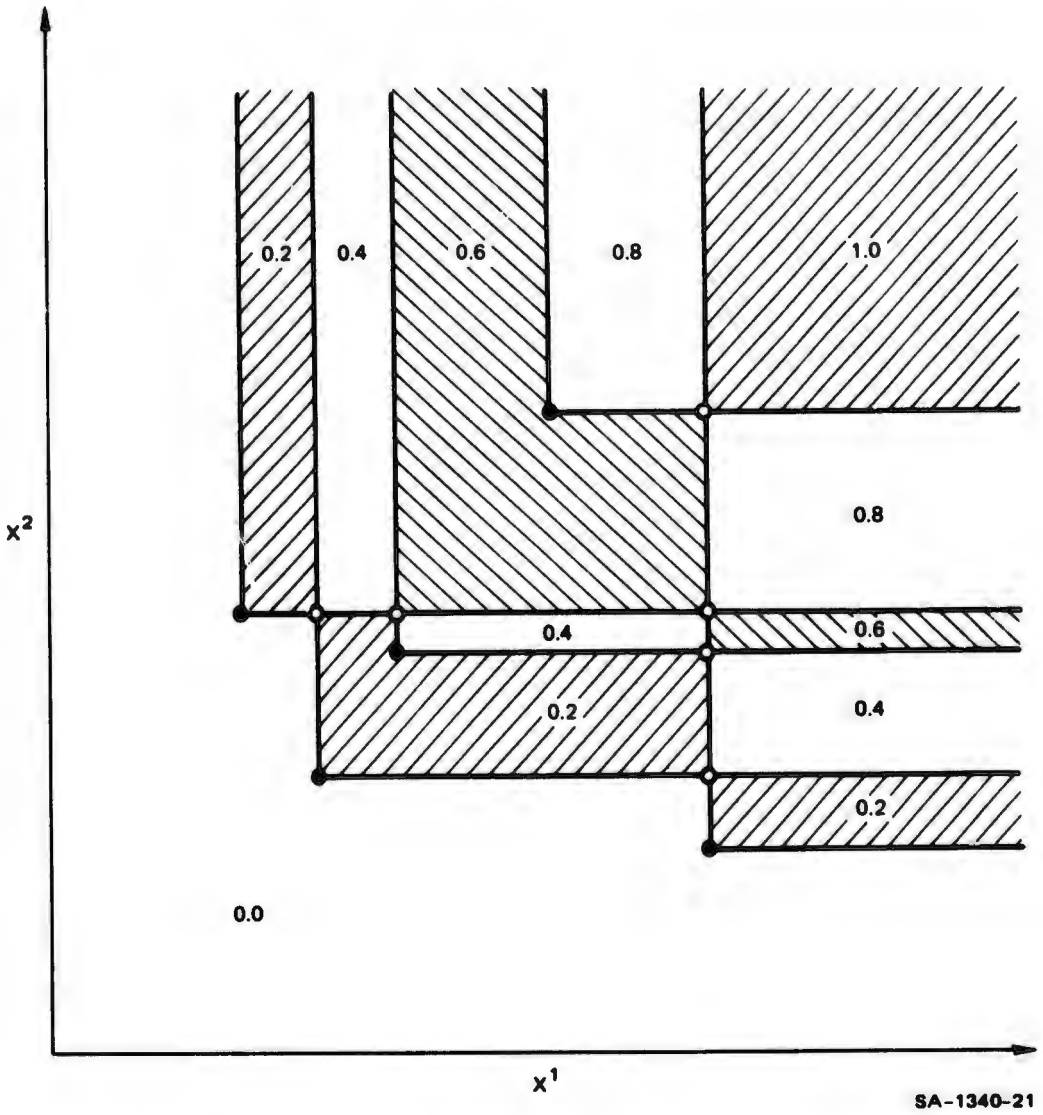


Figure 2. A Two-Dimensional, Empirical Distribution Function for Sample Size = 5

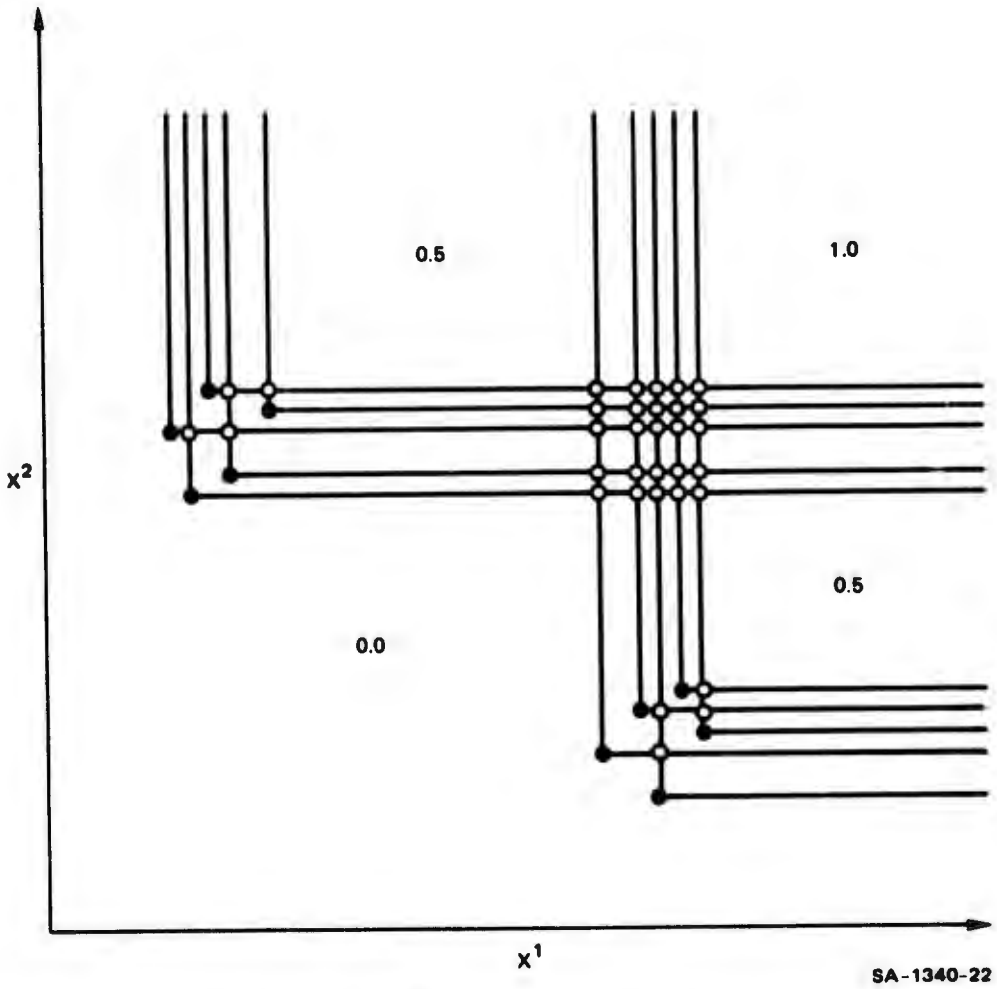


Figure 3. An Empirical Distribution Function for Clustered Data

many more intersection points than data points, which in itself increases the likelihood that the supremum will be found there. In general, the number of intersection points grows as n^d, an exponential growth that effectively prohibits exact computational solutions for any but very small d.

Finally, there is the fact that the multivariate KS test is not distribution-free. This fact was pointed out by Simpson [94], who

suggested the use of one-dimensional projections for which the distribution-free characteristics hold. The KS test is distribution-free for the independent case; in that situation,

$$F(x) = \prod_{j=1}^{d} F_j(x^j)$$

and, for any continuous, univariate distribution functions F_j, the random variables $u_i = F(x_i)$ have the probability density function

$$p(u) = \begin{cases} \frac{1}{(d-1)!}\left[\log_e \frac{1}{u}\right]^{d-1} & 0 \le u \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Note that this distribution varies with dimensionality. For one dimension, the values of $F(x_i)$ are uniformly distributed. As dimensionality increases, it becomes more and more likely that only small values will be observed for $F(x_i)$, since a large value arises only in the unlikely event that $F(x_i^j)$ is large for all j, j = 1,...,d. This is basically why the KS test fails to be distribution-free in general; if there is strong dependence between the variates, the distribution for $F(x_i)$ will not be the same as that for independent distributions of the same dimensionality, but it will resemble the distribution for a lower dimensional distribution.

Actually, the failure of the KS test to be distribution-free is not a serious defect for our application. We are merely trying to test the null hypothesis that the sample came from a normal population. In principle, we can always rotate the coordinates so that they are aligned with the eigenvectors of the covariance matrix, thereby obtaining independent variates if the null hypothesis is true. Alternatively, we can

test the more restrictive hypothesis that the sample came from an independent, normal population. The only truly serious obstacle to using the KS test to determine cluster validity is the exponential growth of computation time with dimensionality. We shall discuss some approximate ways to overcome this problem in Section IV-B-9.

4. The Distribution of Dn(d)

This section considers the effect of dimensionality and sample size on the distribution of $D_n = \sup \left| F_n(x) - F(x) \right|$ when F is known exactly and the variates are statistically independent. In this and in subsequent sections, we obtain our results by Monte Carlo procedures, forming the empirical distribution of D_n for a large number m of samples of the same size n and dimensionality d. The resulting empirical distributions are, of course, only approximations to the true distributions; however, confidence intervals can be derived easily by standard methods.

We estimate the probability $P(\sqrt{n}\, D_n > z)$ by the fraction $\hat{P}$ of the number of samples for which $\sqrt{n}\, D_n$ exceeds z; clearly $1 - \hat{P}$ is the empirical distribution function for $\sqrt{n}\, D_n$. Figure 4 shows the results for d = 1 and n = 4, 25, and 100 using m = 1000 samples. Note that the curve for n = 100 is almost identical to the large-sample theoretical curve shown in Figure 1. The convergence to this limiting curve is quite rapid. The use of the asymptotic results for the finite-sample case is conservative (i.e., leads to less frequent rejection of the null hypothesis), but it would probably be acceptable in most p.r. applications.

In the multivariate case, D_n was computed by evaluating $\left| F_n(x) - F(x) \right|$ at all n^d points that can be formed by exhaustively selecting components of the n d-dimensional sample points. Since evaluation of $F_n(x)$ required n computations, the computation for m samples grew as mn^{d+1}. For large dimensionality, we were computationally limited to

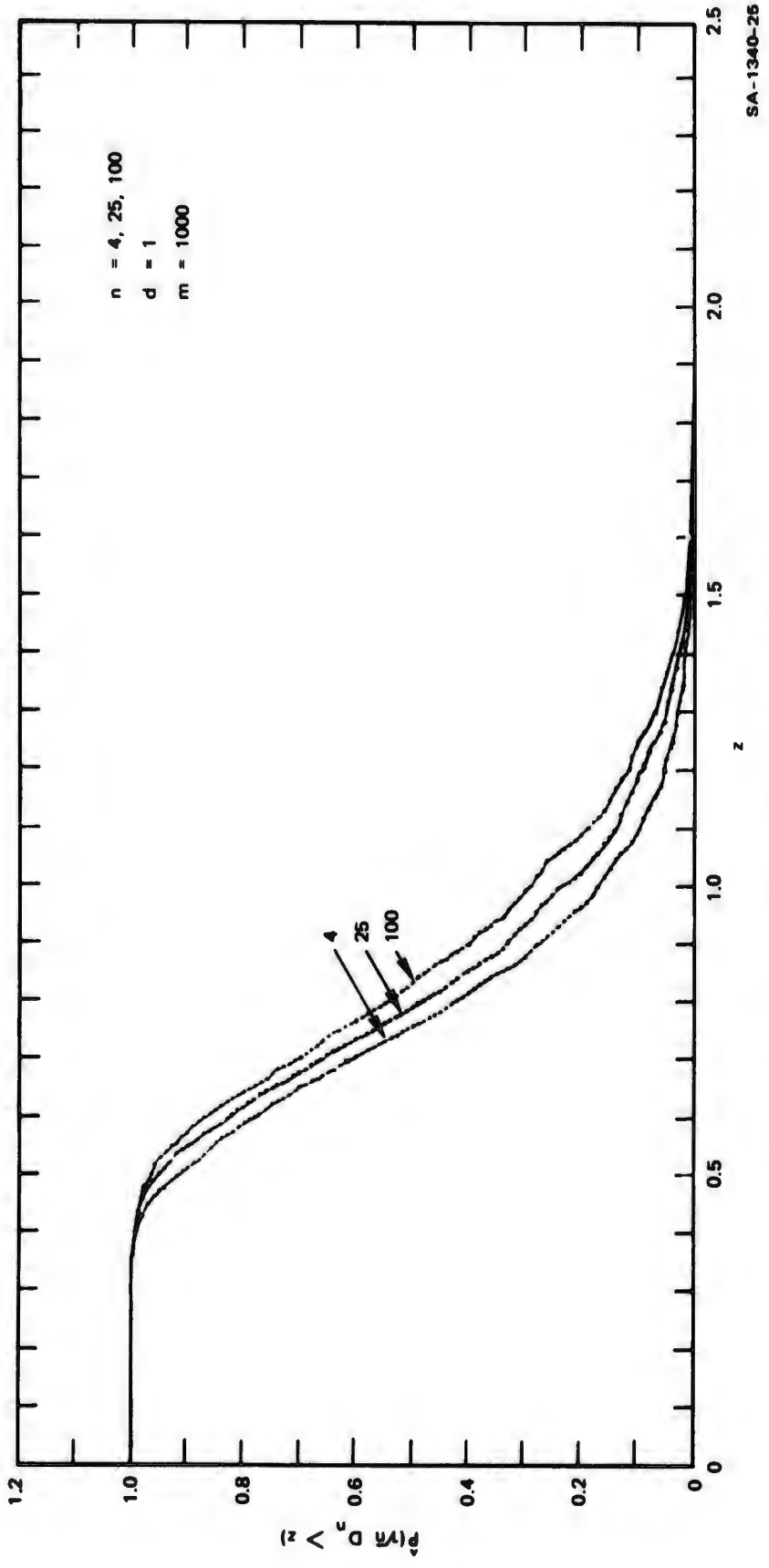


Figure 4. Distribution of D_n for F Known, d = 1

small sample sizes and small numbers of samples. Although we would have preferred to use more samples, m = 100 was the largest practical value for large d, and, for the sake of uniformity, was used throughout.

The empirical distribution functions for $\sqrt{n}\, D_n$ for d ranging from 1 to 5 and for n = 4, 9, 16, and 25 are shown in Figures 5, 6, 7, and 8. In general, these curves show that as the dimensionality increases, one encounters larger values for $\sqrt{n}\, D_n$. For small n and large d (a few points in a high dimensional space), it is not unusual to observe values close to the maximum value, $\sqrt{n}$. Thus, in this case, almost any sample will pass the KS test, and the test fails to reject nonnormal clusters. For large n, the distribution functions presumably approach limiting functions that depend only on dimensionality, but convergence is not as rapid as in the univariate case.

The effects of dimensionality and sample size on $z_{0.05}$, the 5 percent critical value for $\sqrt{n}\, D_n$, are shown in Figure 9. These curves show that $z_{0.05}$ increases monotonically with both d and n. We conjecture that if n is fixed and d is allowed to approach infinity, then these curves approach $\sqrt{n}$. The more interesting problem of finding the large sample critical value is unsolved for d > 1.

5. The Distribution of $\hat{D}_n(d)$

In applying the KS test to determine cluster validity, the distribution function F(x) for the null hypothesis is not known exactly but must be estimated from the sample. If we let $\hat{F}(x)$ be the estimate of F(x) and $\hat{D}_n = \sup \left| F_n(x) - \hat{F}(x) \right|$ be the resulting test statistic, then we want to know the sampling distribution for $\hat{D}_n$. This section presents empirical distributions for $\hat{D}_n$ obtained under the following assumptions:

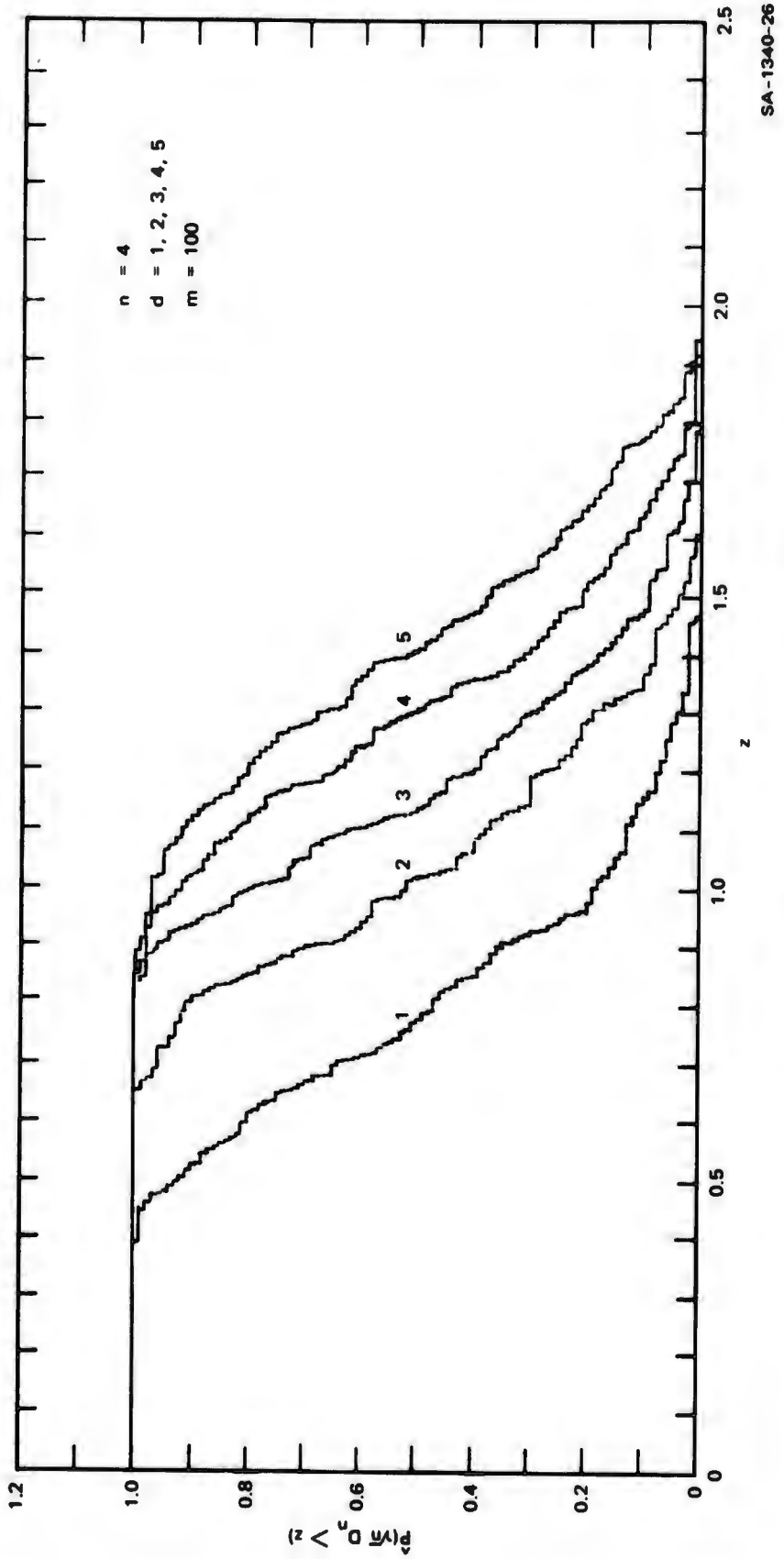


Figure 5. Distribution of D_n for F Known, Sample Size = 4

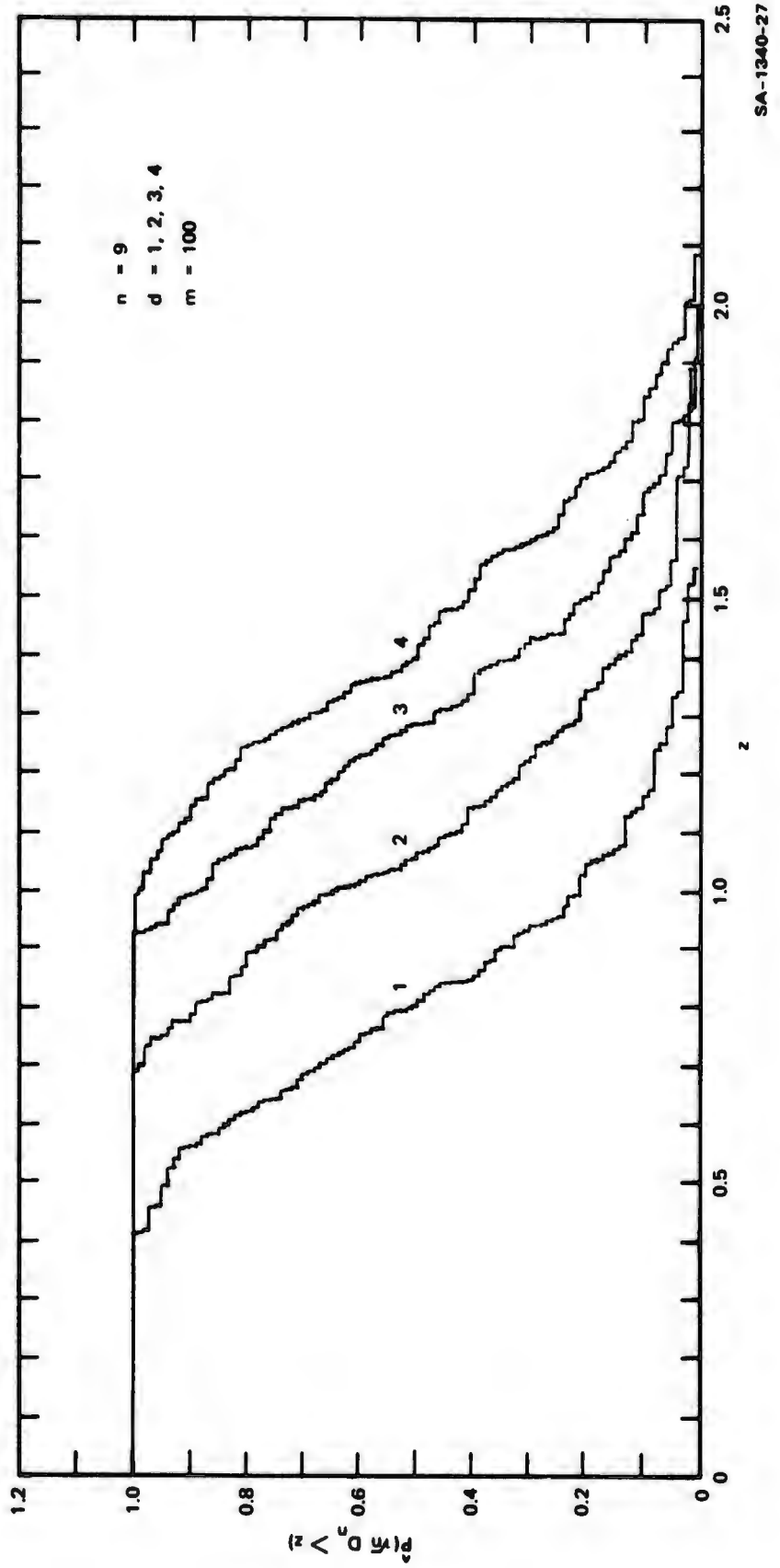


Figure 6. Distribution of D_n for F Known, Sample Size = 9

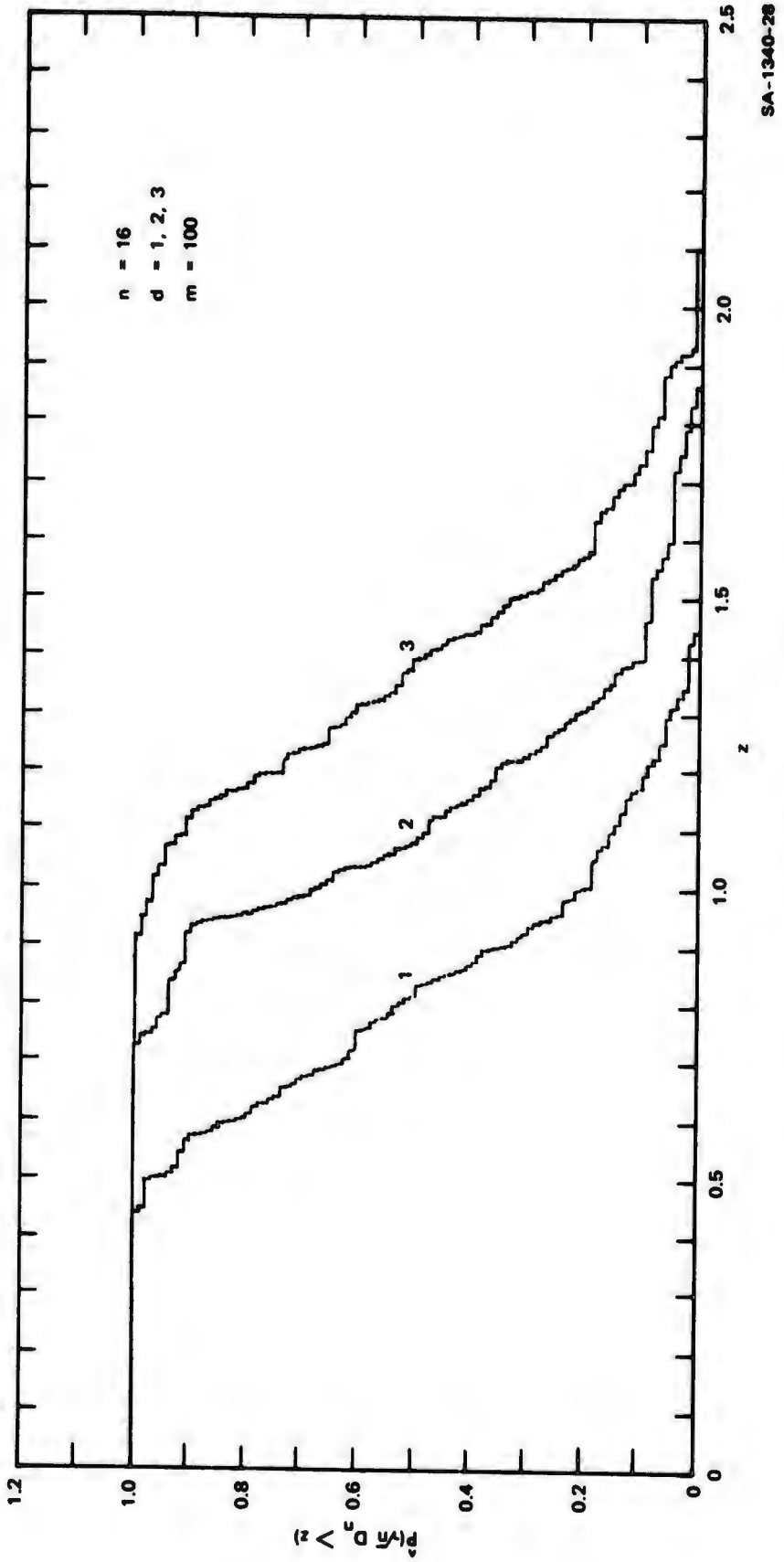


Figure 7. Distribution of D_n for F Known, Sample Size = 16

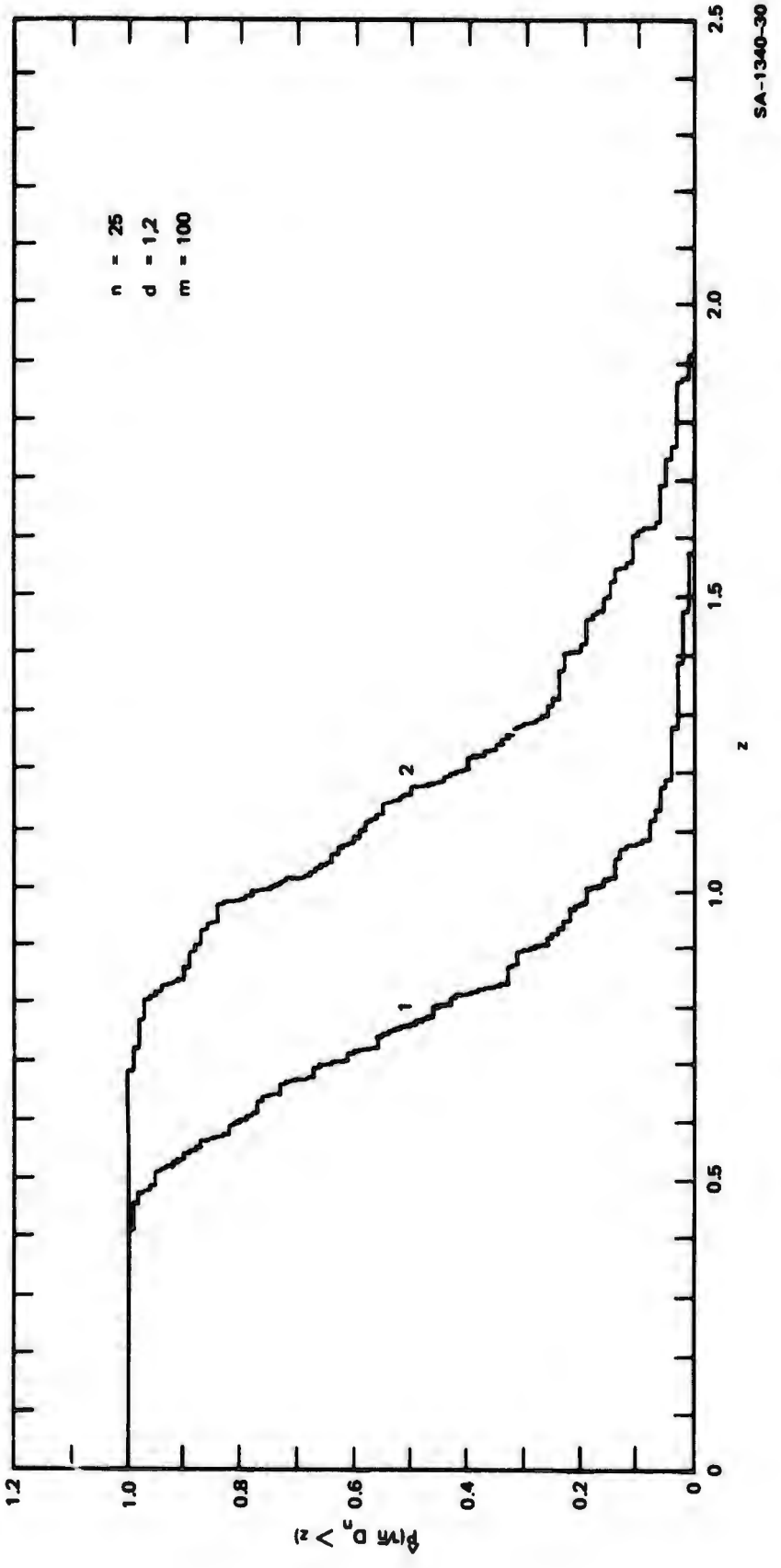


Figure 8. Distribution of D_n for F Known, Sample Size = 25

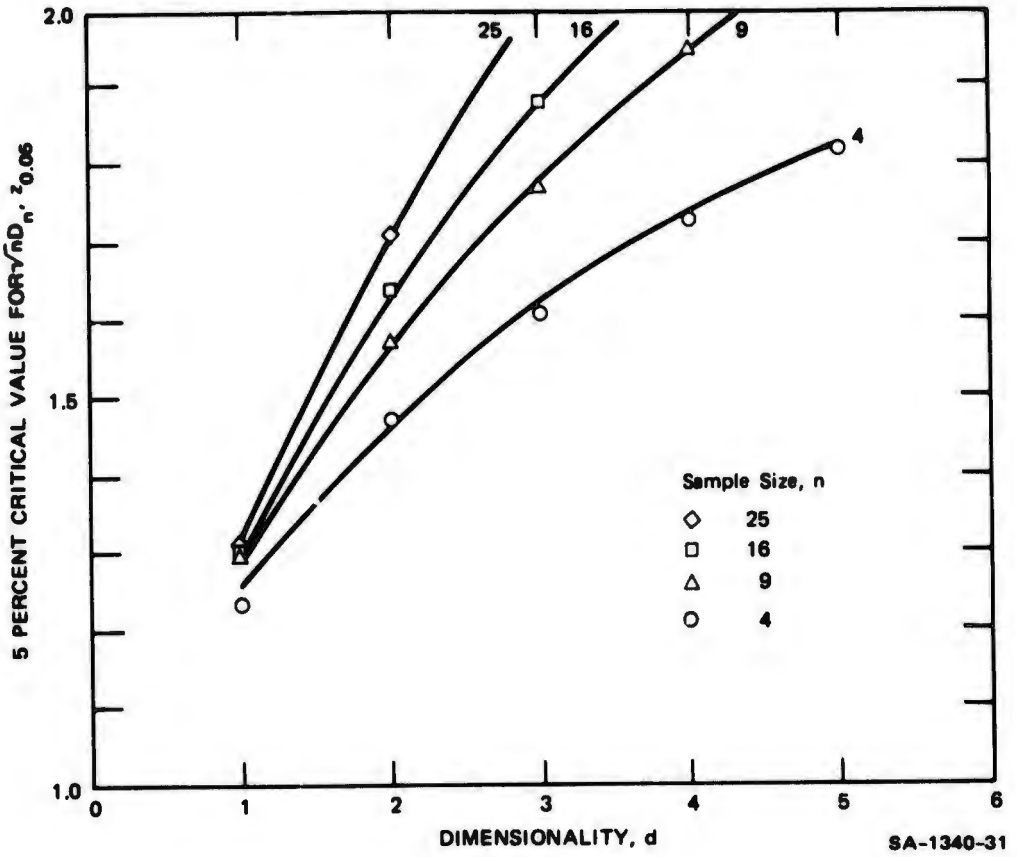


Figure 9. Effect of Dimensionality and Sample Size on 5 Percent Critical Value

- The null hypothesis is that the sample is from an independent normal population with unknown mean and covariance matrix.
- Maximum likelihood estimates are used to estimate the mean and covariance matrix.

The assumption of independence is a strong one that is not likely to be satisfied in practice and thus deserves some comment. As we observed in Section IV-B-3, with this assumption, the KS test remains distribution-free, the distribution of D_n depending only on the sample size n and the dimensionality d. Monte Carlo experiments for determining this distribution then have only these two variables with which to contend. If the distribution function F(x) were multivariate normal and completely known, one could always rotate the coordinate axes to coincide with the

eigenvectors of the covariance matrix, thereby obtaining independent variates. In the large-sample case, it is plausible that the sample covariance matrix can be substituted for the true covariance matrix with equivalent results. In the small-sample case, sampling variations undoubtedly effect the change in coordinates and modify the distribution of the KS statistic. However, it should be remembered that we are primarily interested in detecting fairly gross departures from multivariate normality, and this procedure should be generally useful for this purpose.

Figure 10 shows the empirical distribution for $\sqrt{n}\,\hat{D}_n$ for d = 1 and n = 4, 9, 16, 25, and 100 using m = 1000 samples. For large n, these curves agree closely with the results given by Lilliefors [93]. The relatively small, though statistically significant, differences that occur for n = 4 and n = 9 are probably due to the fact that Lilliefors divided by n-1 rather than n in estimating the variance. If the curves in Figure 10 are compared with the corresponding curves in Figure 4, one sees that critical values for $\hat{D}_n$ are roughly two-thirds to three-quarters of the critical values for D_n. Thus, when the sample is used to establish the unknown parameters, the estimate $\hat{F}(x)$ always provides a significantly better approximation to the empirical distribution function $F_n(x)$ than does the true distribution function F(x).

A similar set of curves for the bivariate independent case is shown in Figure 11. For d = 2, the 5 percent critical values for $\sqrt{n}\,\hat{D}_n$ are approximately 0.73 ± 0.02 times the corresponding critical values for $\sqrt{n}\,D_n$. The curves for d from 1 to 5, n = 4, and m = 100 shown in Figure 12 also yield this relation. Thus, a suggested rule of thumb for obtaining 5 percent critical values for $\sqrt{n}\,\hat{D}_n$ is to multiply the values for $\sqrt{n}\,D_n$ given in Figure 9 by 0.73.

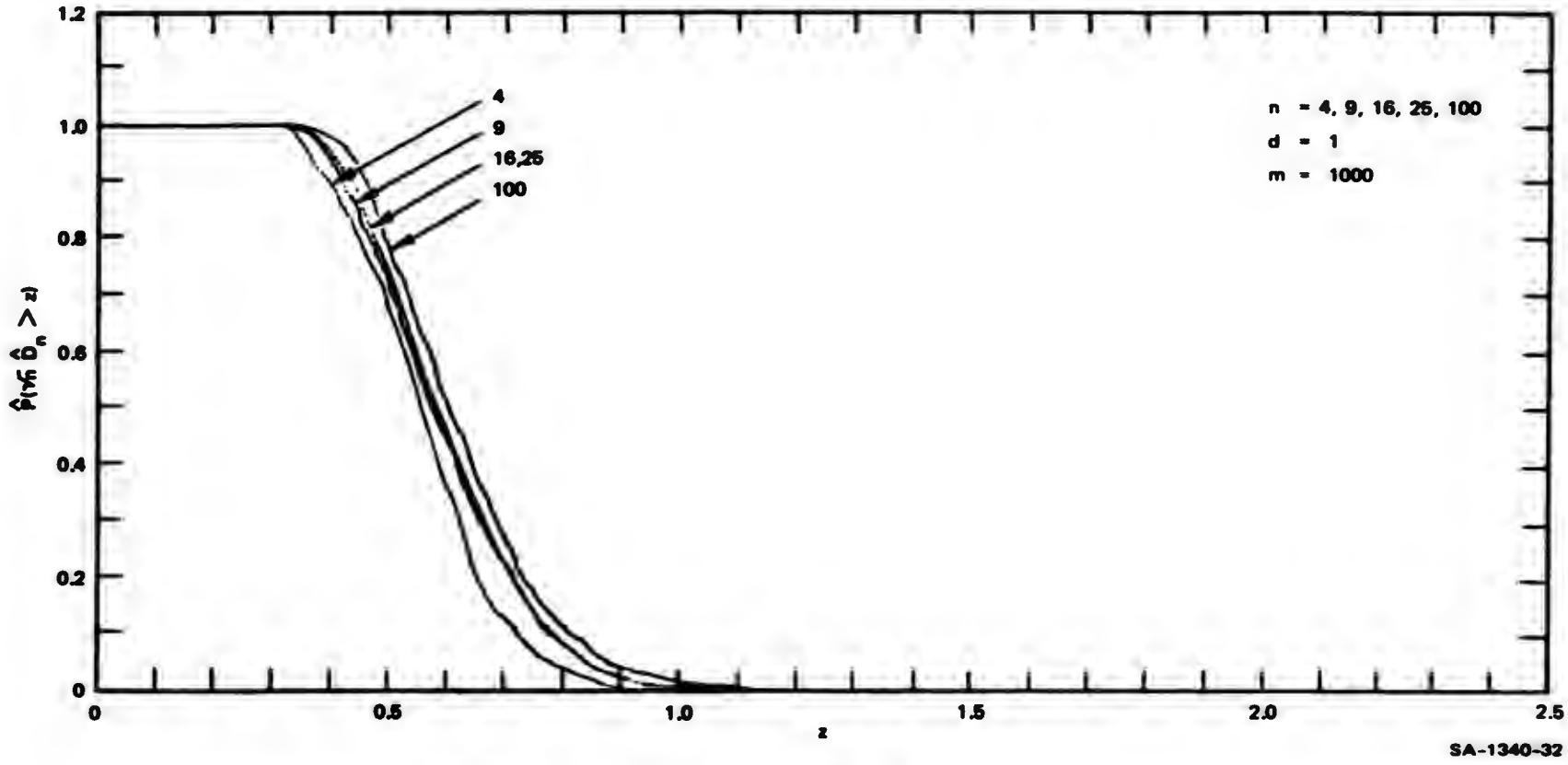


Figure 10. Distribution of $\hat{D}_n$, Dimensionality = 1

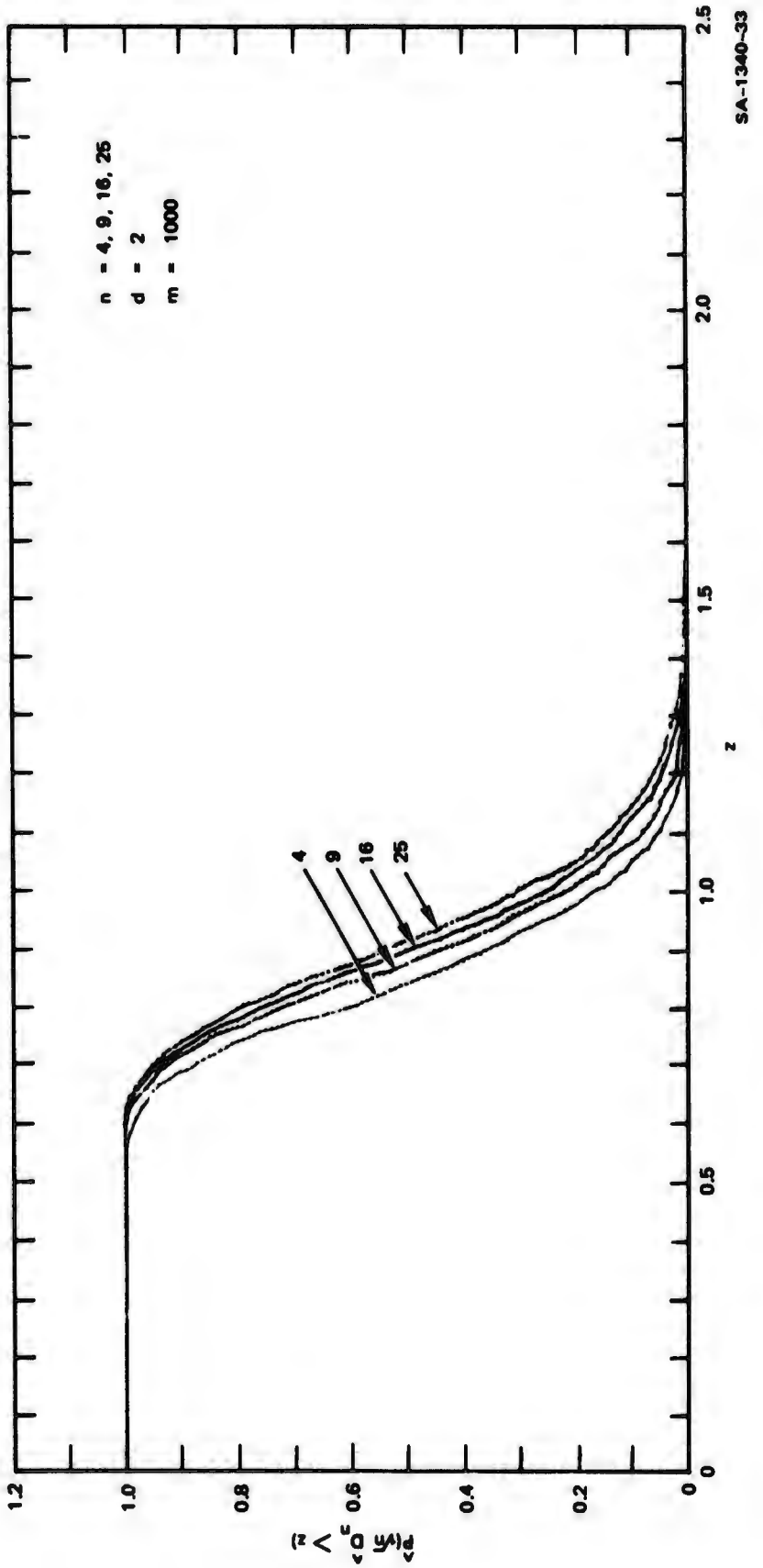


Figure 11. Distribution of $\hat{D}_n$, Dimensionality = 2

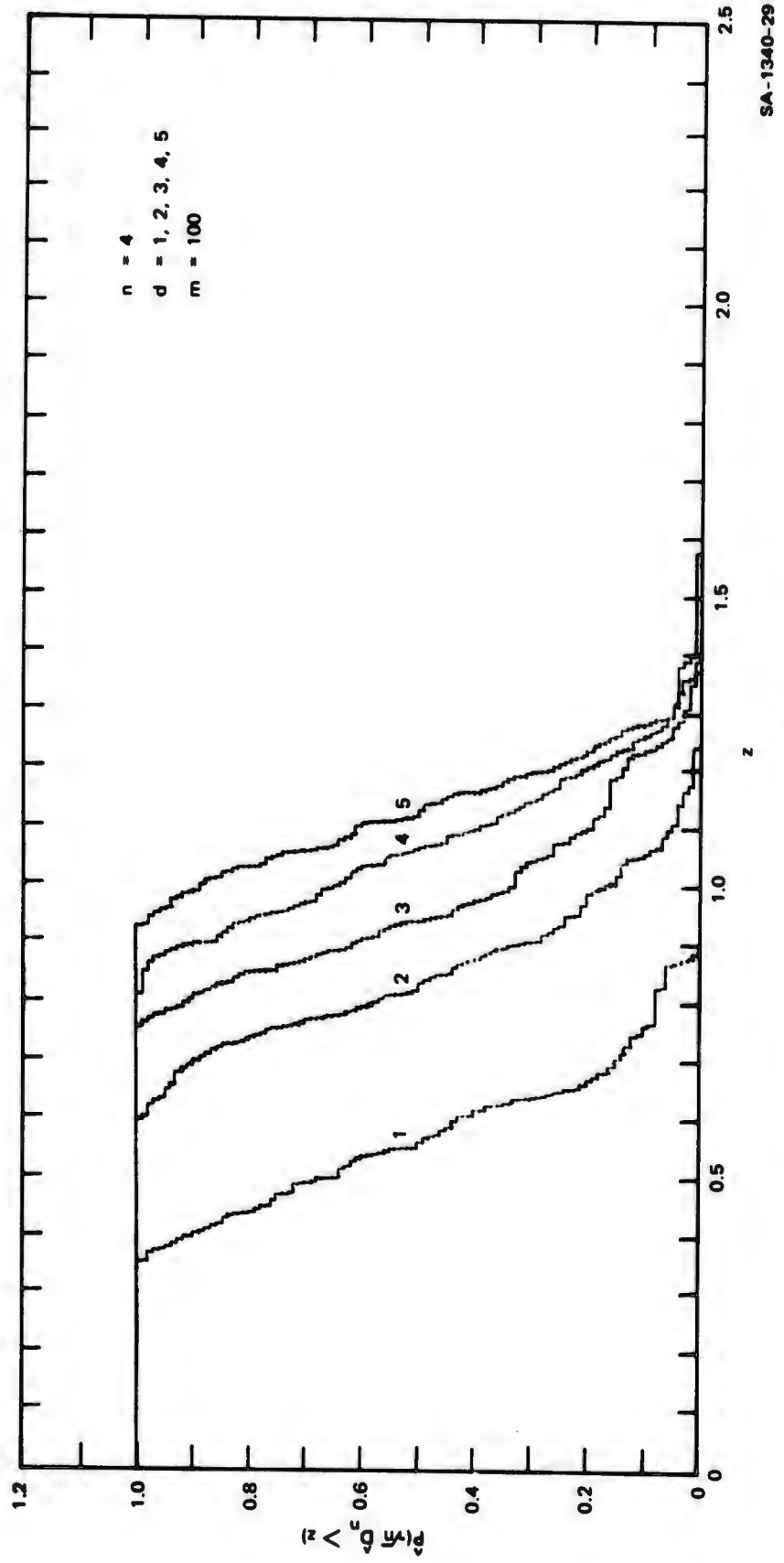


Figure 12. Distribution of $\hat{D}_n$, Sample Size = 4

6. Application to Ceiling Height Data

To illustrate the use of these results, we applied them to two sets of real data. The first data set consisted of a series of measurements of cloud base heights obtained from a single rotating-beam ceilometer. These data were collected by the U.S. Department of Commerce National Weather Service and were used in an earlier SRI study of automatic measurement of cloud height and amount at airports (Duda et al., [95]).

When several layers of clouds are present, successive base-height measurements can experience large changes as different layers are detected. Smaller changes are usually due to the normal irregularity of the cloud base. To process such data, the readings must be separated or clustered according to cloud layer, so that statistics such as the average ceiling height will be meaningful. In an automated instrumentation system, a clustering algorithm must replace the human observer who normally interprets the instrument readings.

The data plotted in Figure 13 show a portion of a ceilometer record during which two or more cloud layers were being detected. A high layer around 1500 feet was intermittently detected through breaks in lower clouds ranging from about 100 to 400 feet. The histogram shows that the upper layer was quite well defined, but it is not clear whether the lower clouds exist in several distinct layers or one erratic, undulating layer. Thus, an important problem is to determine how many layers are present.

There are many ways in which one can use the KS test to solve this problem. For simplicity, we applied an elementary hierarchical clustering procedure to the collection of n = 67 ceiling height readings, taking no account of their temporal sequence. The clustering procedure started with all 67 readings in one cluster. Clusters were successively split into two parts until they passed the KS test at the 5 percent level.

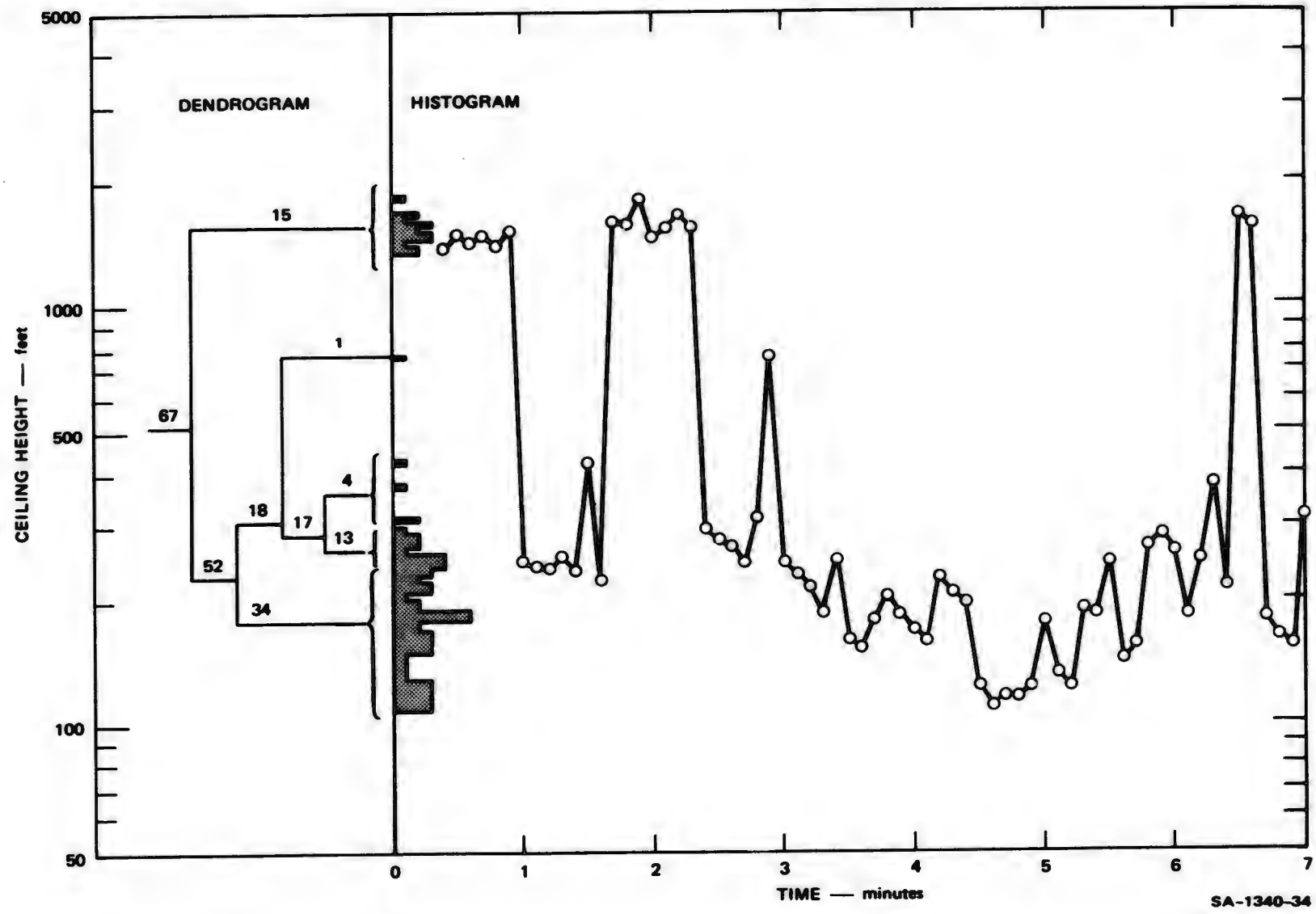


Figure 13. Ceiling Height Data, Sterling, Virginia, 3 February 1970

Splitting was done by an elementary ISODATA algorithm. For each cluster to be split, two initial cluster centers, m_1 and m_2, were selected, m_1 starting as the sample mean for the cluster and m_2 starting as the point at which the maximum value for $\left| F_n(x) - \hat{F}(x) \right|$ was achieved. The cluster was split into two parts, Υ_1 and Υ_2, by assigning a point x to Υ_1 if $\|x - m_1\| < \|x - m_2\|$ and to Υ_2 otherwise. The cluster centers at iteration k+1 were then computed as the sample means for Υ_1 and Υ_2 at iteration k, and the process was repeated until no further changes occurred.

The effect of this procedure is summarized in the dendrogram in Figure 13. For the initial cluster of all 67 data points, the value of 3.02 for $\sqrt{67}\ \hat{D}_{67}$ was far above the 0.87 critical value (obtained directly from Figure 10), and the cluster was split. One part consisted of the 15 readings around 1500 feet. This cluster passed the KS test with $\sqrt{15}\ \hat{D}_{15} = 0.44$, and was not subdivided further. The other part consisted of the low altitude readings, plus a sprinkling of higher readings going up to 760 feet. These 52 readings failed the KS test with $\sqrt{52}\ \hat{D}_{52} = 1.22$, and this cluster was also split. The part consisting of the 34 readings up to 240 feet passed the KS test with $\sqrt{34}\ \hat{D}_{34} = 0.49$. The remaining 18 readings from 240 to 760 feet were further subdivided, leading eventually to three small clusters of doubtful meteorological significance. However, all of them at least passed the KS test for normality. Moreover, by removing these clusters, much more meaningful values were obtained for the sample mean and variance for the important lowest layer.

7. Application to Iris Data

The second experiment used a set of physical measurements of the flowers of fifty cases each of three species of iris collected by E. Anderson and listed in a classical paper by R. A. Fisher [21].

This data set is important chiefly because it has been used as a test example in many papers on clustering. In our experiments, we used only the petal length and petal width measurements, which produced a sample of size n = 150 and dimensionality d = 2.

The data listed by Fisher are limited to two-significant-digit accuracy, and when only the length and width measurements are used, many of the data points coincide. These ties cause large jumps in the empirical cumulative distribution function, leading to needless rejection of the normal null hypothesis. To prevent this from happening, we added to the original data zero-mean random noise, uniformly distributed over the 0.1 cm quantization interval. The resulting data points are shown in Figure 14. The species Iris Setosa is clearly separated from the other two species, Virginica and Versicolor, and the distributions for all three species appear roughly normal.

The same elementary hierarchical clustering procedure described in Section IV-B-6 was used to cluster these data. The only changed needed was to increase the 5 percent critical value for the two-dimensional case to 1.23, a value obtained directly from Figure 11. The resulting hierarchical partitioning of the data set is shown in Figure 14. For the entire set of 150 data points, we obtained $\sqrt{150}\,\hat{D}_{150} = 3.69$, which was well above the 5 percent critical value. Splitting produced two sets, indicated as G1 and G2 in Figure 14. G1 contained all but one of the 100 Virginica and Versicolor points, and G2 contained all 50 Setosa points plus the stray Versicolor point. This one point caused G2 to fail the KS test with $\sqrt{51}\,\hat{D}_{51} = 2.24$. Its subsequent subdivision into G5 and G6 was not done very well, and another splitting of G6 into G9 and G10 was required to produce three subsets that passed the KS test. The values of the KS statistic for G5, G9, and G10 were 1.10, 0.79, and 0.75, respectively.

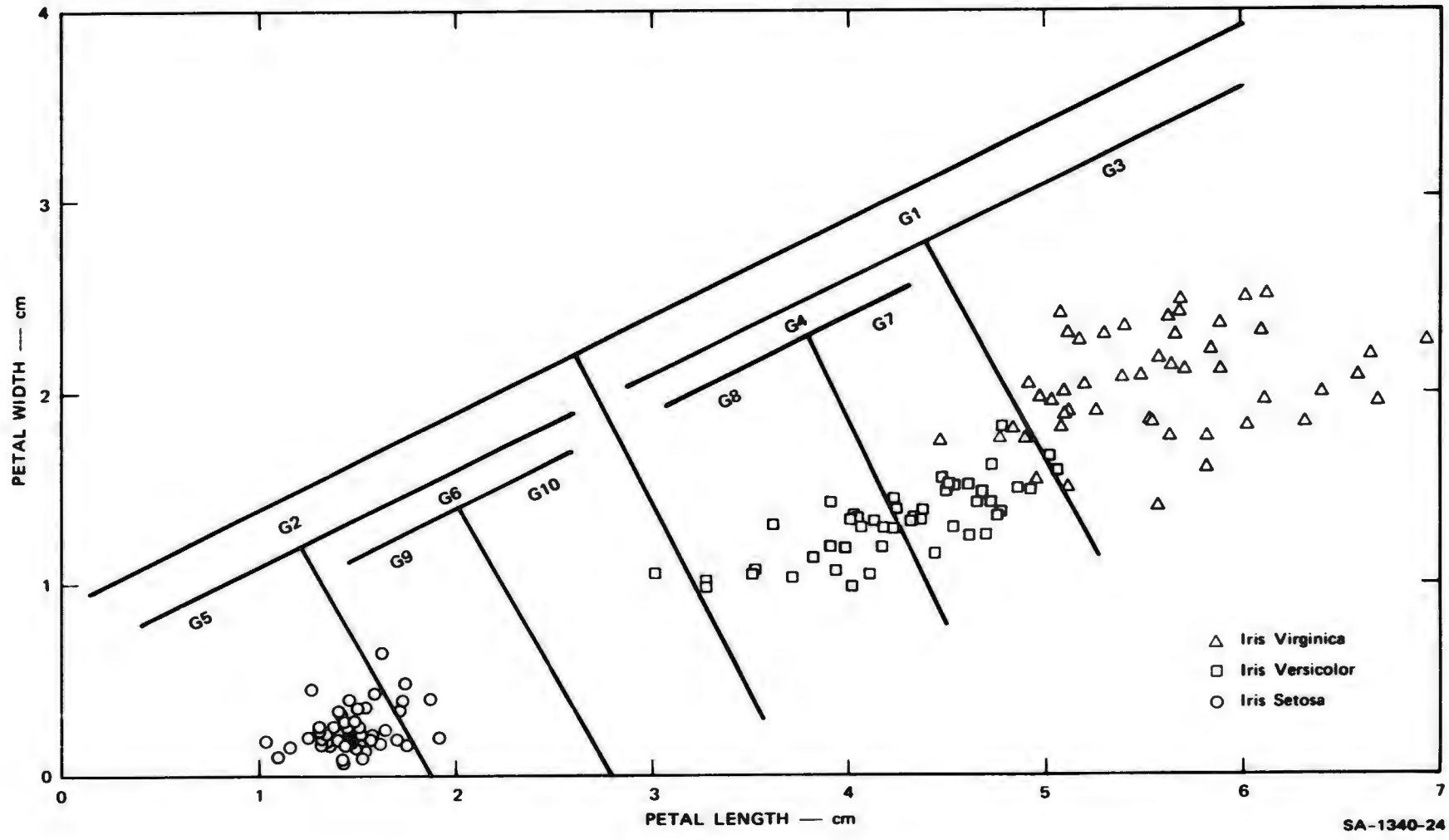


Figure 14. The Anderson Iris Data

The value of the KS statistic for G1--the mixture of Virginica and Versicolor--was 2.76, and G1 was split into G3 and G4. This division appears to yield a very good separation of the two species, and the 46 points in G3 did pass the KS test with $\sqrt{46}\,\hat{D}_{46} = 1.00$. However, the 53 points in G4 just failed to pass with $\sqrt{53}\,\hat{D}_{53} = 1.37$. When G4 was split into G7 and G8, both of these sets passed with values 1.16 and 0.98, respectively.

These examples illustrate how the KS test can be used to establish cluster validity. We do not claim that the partitionings obtained are optimal, or even necessarily good. The hierarchical procedure for splitting clusters was very rudimentary, and the partitions formed were not always proper. However, the KS test did appear to provide a proper evaluation of the partitionings; i.e., when an alleged cluster failed the test, there were always good reasons to suspect that it was not a single, normal cluster, and when it passed, a normal cluster description always appeared to be reasonable. Used in conjunction with more sophisticated clustering procedures, the KS test provides a valuable way to test cluster validity.

8. Type II Error for Univariate Normal Mixtures

So far we have considered only the Type I error rate, the probability of rejecting the normal hypothesis when the sample is drwan from a normal population. This section briefly considers the Type II error rate, the probability of accepting the normal hypothesis when the sample is not drawn from a normal population. Of course, this error rate depends on the probability law for the population and how much it differs from the normal law. The nonnormal distribution we consider is the simple univariate mixture density

$$p_x(x) = \frac{1}{2\sqrt{2\pi}\,\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma}\right)^2\right] + \frac{1}{2\sqrt{2\pi}\,\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma}\right)^2\right]$$

By introducing the normalized variable

$$u = \frac{x - \frac{1}{2}(\mu_1 + \mu_2)}{\sigma}$$

and the normalized separation

$$\mu = \frac{\mu_2 - \mu_1}{2\sigma}$$

we can just as well work with the simpler density

$$p_u(u) = \frac{1}{2\sqrt{2\pi}}\left\{\exp\left[-\frac{1}{2}(x+\mu)^2\right] + \exp\left[-\frac{1}{2}(x-\mu)^2\right]\right\}$$

This density, which is sketched in Figure 15, has mean 0 and variance $1 + \mu^2$. If μ is less than 1, it is unimodal and closely resembles a normal density. If μ is greater than 1, it is bimodal, and for μ greater than about 3, it corresponds to a population exhibiting two distinct, well-separated clusters.

The ability of the KS test to discriminate between this density and a normal desnity depends on the sample size and on the difference

$$D(\mu) = \sup_u \left|F(u;\mu) - F_N(u;\mu)\right|$$

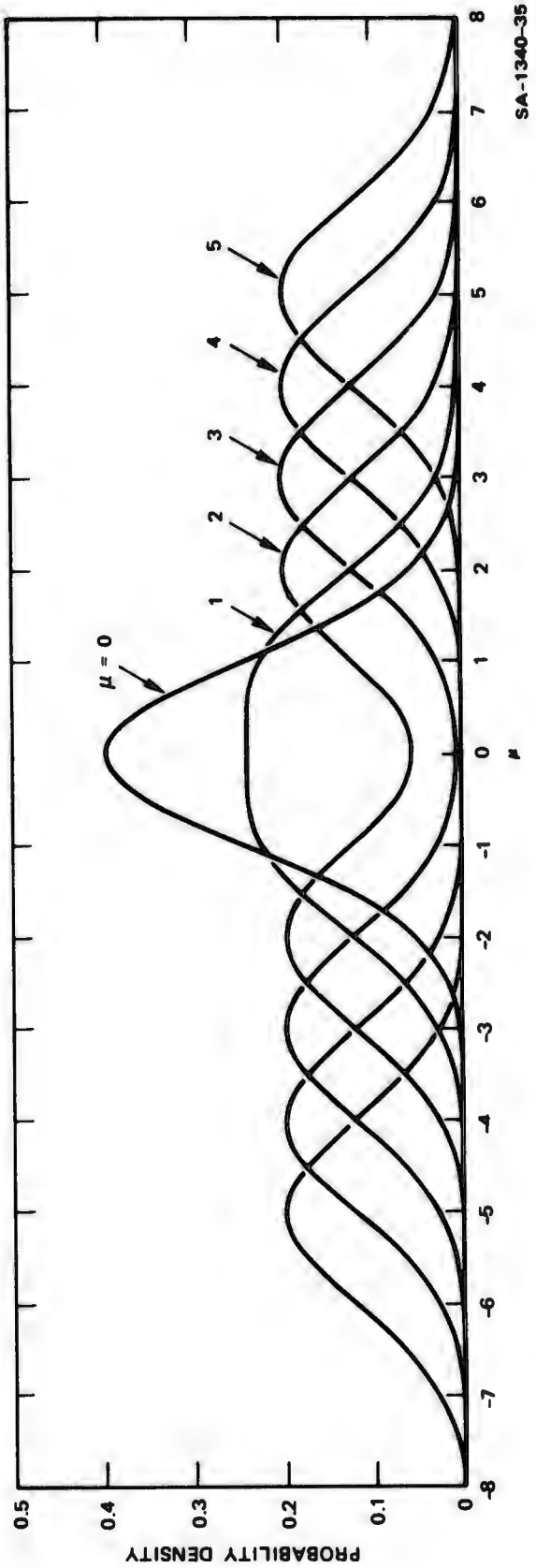


Figure 15. A Mixture of Two Univariate Normal Densities

where $F(u;\mu)$ is the distribution function for $p_u(u)$, and $F_N(u;\mu)$ is the distribution function for the normal distribution $N(0,1+\mu^2)$. If we write

$$\Phi(u) = \int_{-\infty}^{u} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

then

$$F(u;\mu) = \frac{1}{2}\left[\Phi(u-\mu) + \Phi(u+\mu)\right]$$

and

$$F_N(u;\mu) = \Phi\left(\frac{-u}{\sqrt{1+\mu^2}}\right)$$

The function $D(\mu)$ is shown in Figure 16. If μ is less than one, $D(\mu)$ is quite small, implying that it is quite difficult to tell the difference between the mixture and a single, "best-fitting" normal density. $D(\mu)$ increases monotonically with μ, approaching the asymptotic value of $\Phi(1) - \Phi(0) = 0.3413$ as μ approaches infinity.

In theory, if $D(\mu)$ is not zero and if the sample size is sufficiently large, one can always discriminate between the mixture and the single density. A rough estimate of the required sample size can be obtained by assuming that the KS statistic $\hat{D}_n$ is related to the difference D by

$$\hat{D}_n = D + e$$

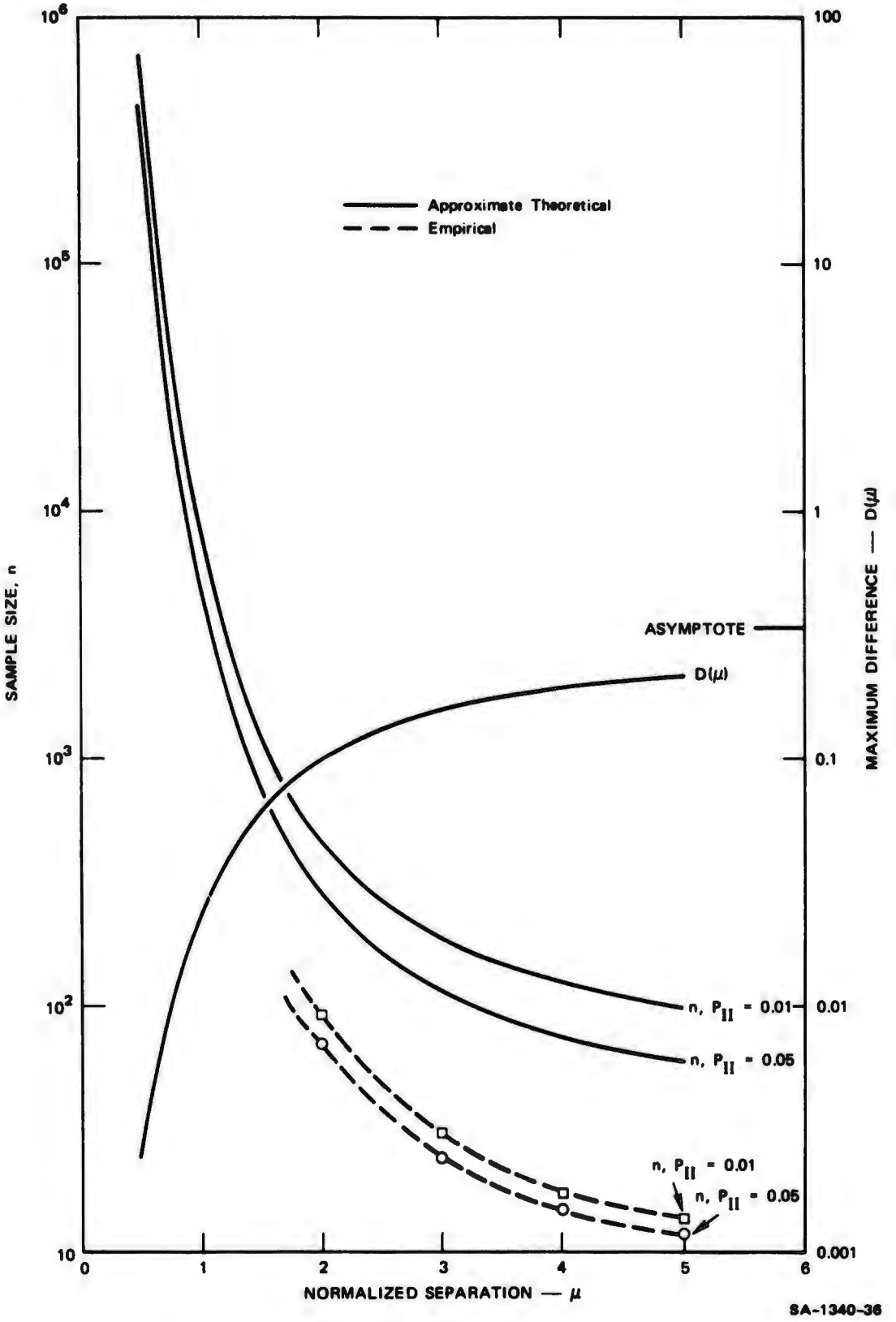


Figure 16. The Effect of Separation on the Distinguishability of a Normal Mixture

where the distribution of the error e is assumed to be $N[0,F(1-F)/n]$, $F(u;\mu)$ being evaluated at the u that leads to the maximum D; i.e., we are assuming that the maximum difference of the empirical distribution function and the "best-fitting" normal distribution function occurs where the population distributions are most distant, using the asymptotic normal approximation to account for sampling error in estimating $F(u;\mu)$ and neglecting sampling error in estimating $F_N(u;\mu)$. It follows that the distribution of $\sqrt{n}\,\hat{D}_n$ under the mixture hypothesis is approximately normal, with mean $\sqrt{n}\,D(\mu)$ and variance $F(1 - F)$.

If the Type I error probability is set by the critical value z_α for $\sqrt{n}\,\hat{D}_n$, then the Type II error probability P_{II} is given approximately by

$$P_{II} \approx \Phi\left(\frac{z_\alpha - \sqrt{n}\,D(\mu)}{\sqrt{F(1 - F)}}\right)$$

This equation can be used either to determine P_{II} or to determine the sample size needed to achieve a specified Type II error rate. Figure 16 shows the required sample size for $z_\alpha = 0.87$ (obtained from Figure 10 for a 5 percent Type I error rate) for 1 percent and 15 percent Type II error rates. As expected, the required sample size is extremely large for $\mu \leq 1$, where the mixture distribution closely resembles a normal distribution. In fact, because of the approximate nature of the analysis, the sample size requirements generally exceed what is actually necessary, but the variation of n with μ is at least qualitatively correct.

To determine the Type II error rate more exactly, a series of Monte Carlo experiments were performed using m = 1000 univariate samples of size n = 4, 10, 16, 25, and 100. The resulting empirical distributions for $\hat{D}_n$ are shown in Figures 17 to 21 for μ = 1 to 5, respectively. For

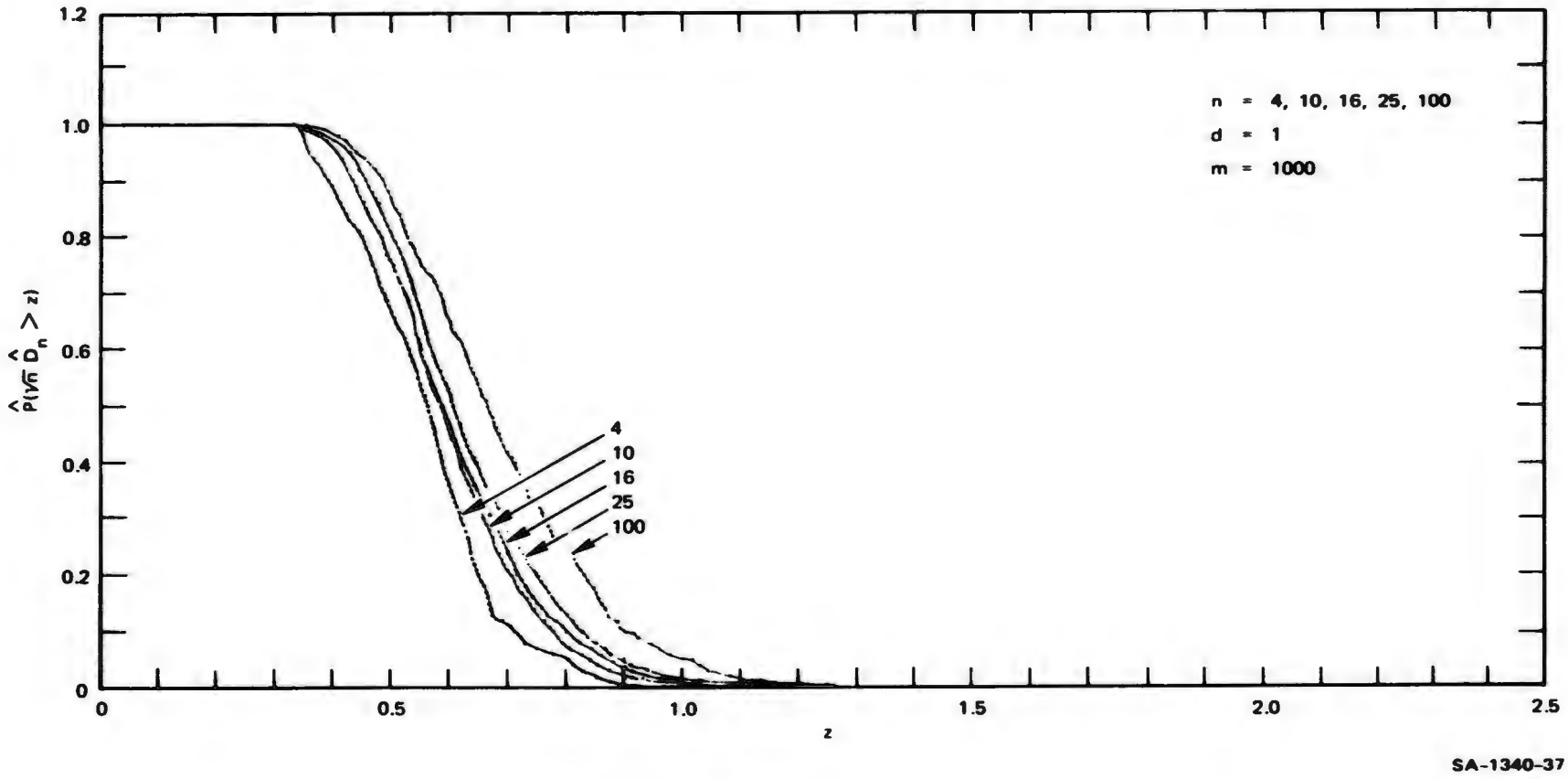


Figure 17. Distribution of $\hat{D}_n$ Under Mixture Hypothesis, $\mu = 1$

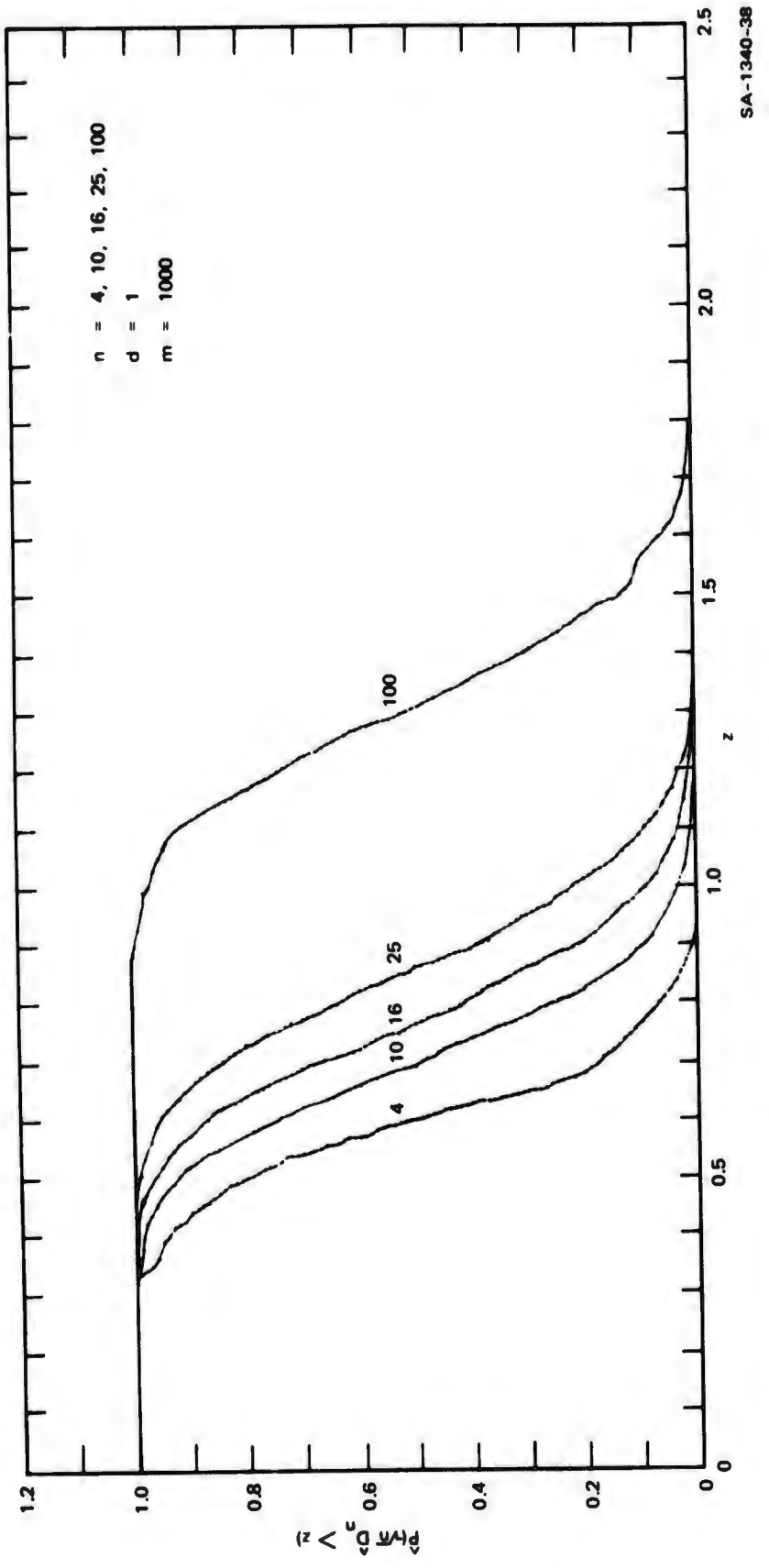


Figure 18. Distribution of $\hat{D}_n$ Under Mixture Hypothesis, $\mu = 2$

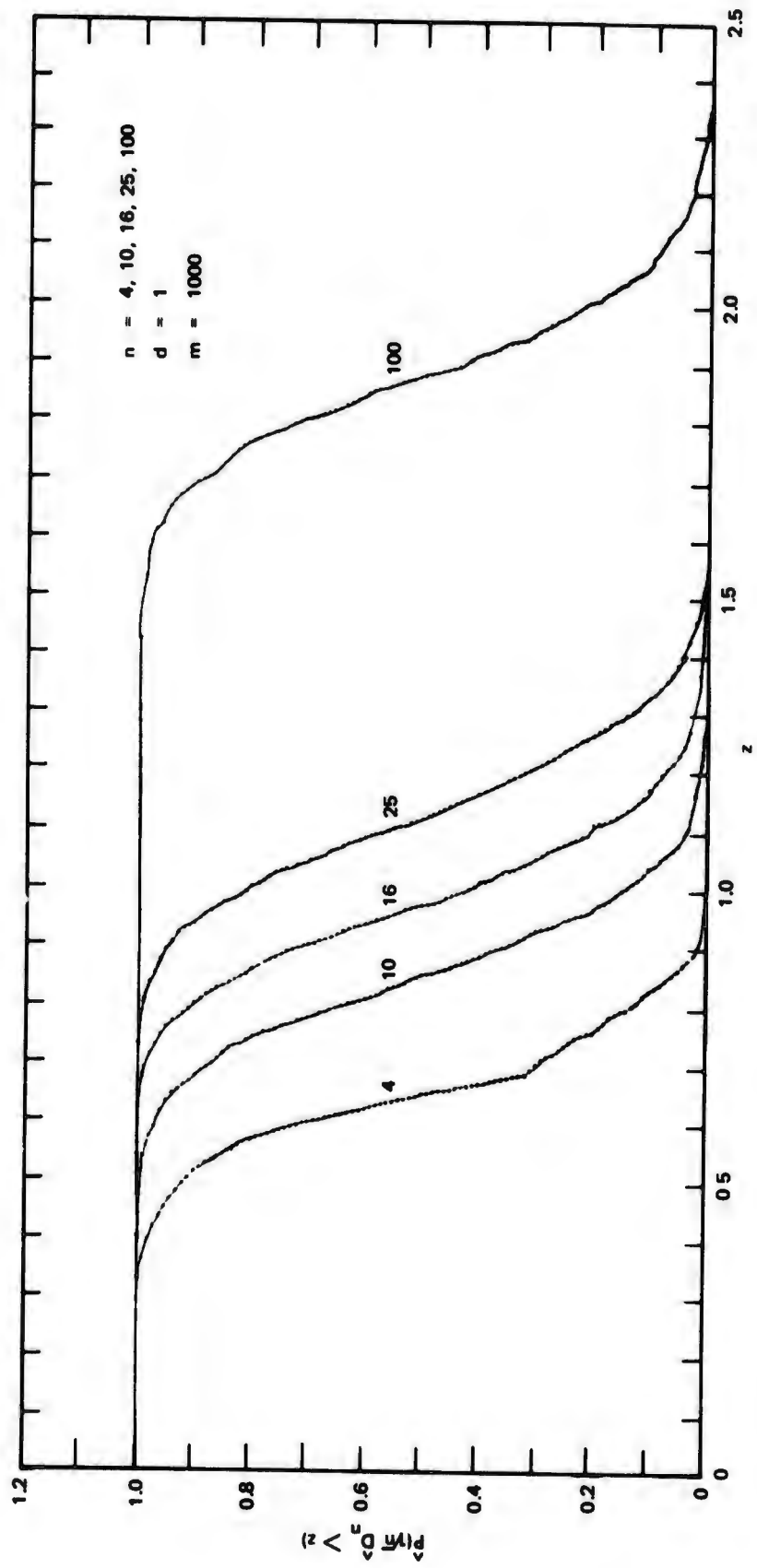


Figure 19. Distribution of $\hat{D}_n$ Under Mixture Hypothesis, $\mu = 3$

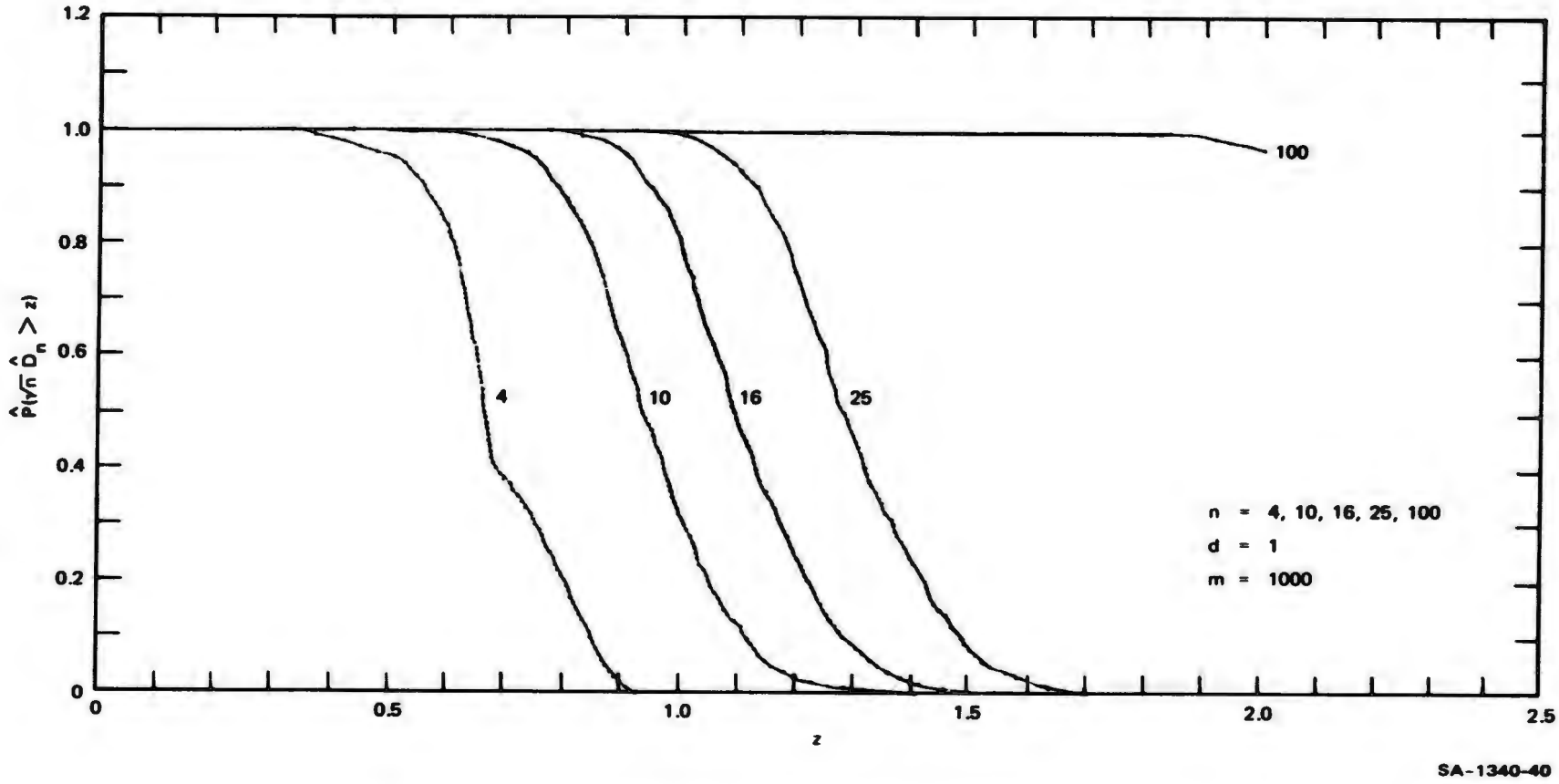


Figure 20. Distribution of $\hat{D}_n$ Under Mixture Hypothesis, $\mu = 4$

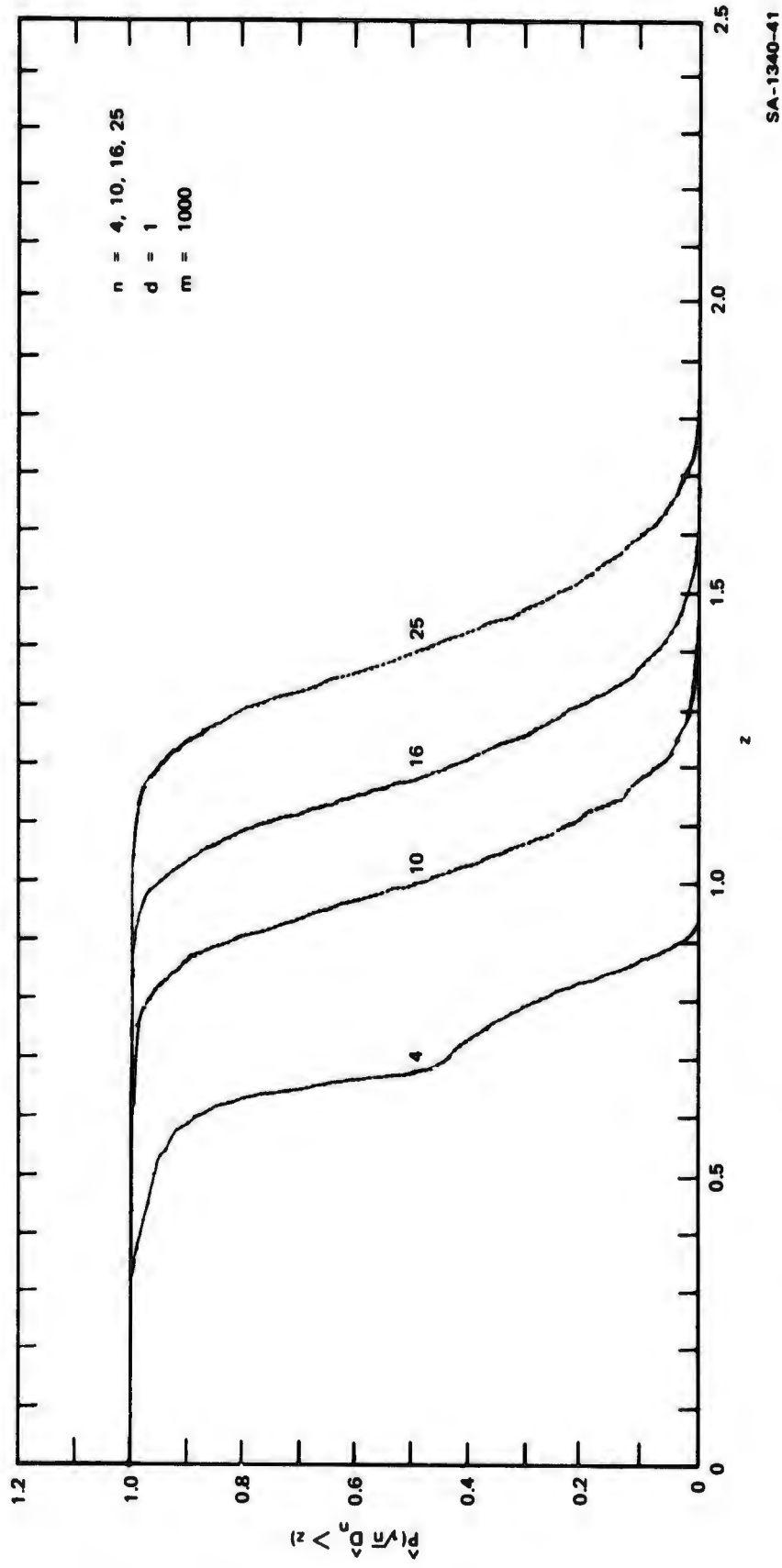


Figure 21. Distribution of $\hat{D}_n$ Under Mixture Hypothesis, $\mu = 5$

$\mu = 1$, the results are very similar to those shown in Figure 10 for the normal case. Unless the sample size is very large, it is quite difficult to distinguish between the resulting mixture and a single, normal distribution. Even for n = 100, the Type II error rate for a 5 percent Type I error rate ($z_{0.05} = 0.87$) is 87 percent.

As μ increases, the Type II error rate decreases. For example, for n = 100, it drops below 1 percent for $\mu = 2$. Interestingly enough, even if μ is quite large, the Type II error rate may remain large if the sample size is small. Basically, this is because there is an appreciable probability with a small sample that most of the individuals will come from just one of the two components. This phenomenon is also reflected in the shapes of the curves for n = 4, which exhibit noticeable slope changes corresponding to the 2/2, 3/1, and 4/0 possible splits between the two components.

These curves can be used to obtain the Type II error rate once the sample size n, the normalized separation μ, and the critical value z_α are known. Figure 22 gives P_{II} as a function of n and μ for the 5 percent critical value $z_{0.05} = 0.87$. The alternative curves of n as a function of P_{II} and μ are shown in Figure 16. The general conclusion is that the KS test can effectively separate a bimodal mixture from a unimodal cluster with a relatively small sample size, provided that the components of the mixture are far enough apart to give two well-defined clusters.

9. Remarks

The Kolmogorov-Smirnov test provides a means for evaluating the results of any clustering procedure intended more or less for a mixture of normal distributions. It is a test for validity that measures directly how well a description given in terms of normal clusters actually fits the data at hand.

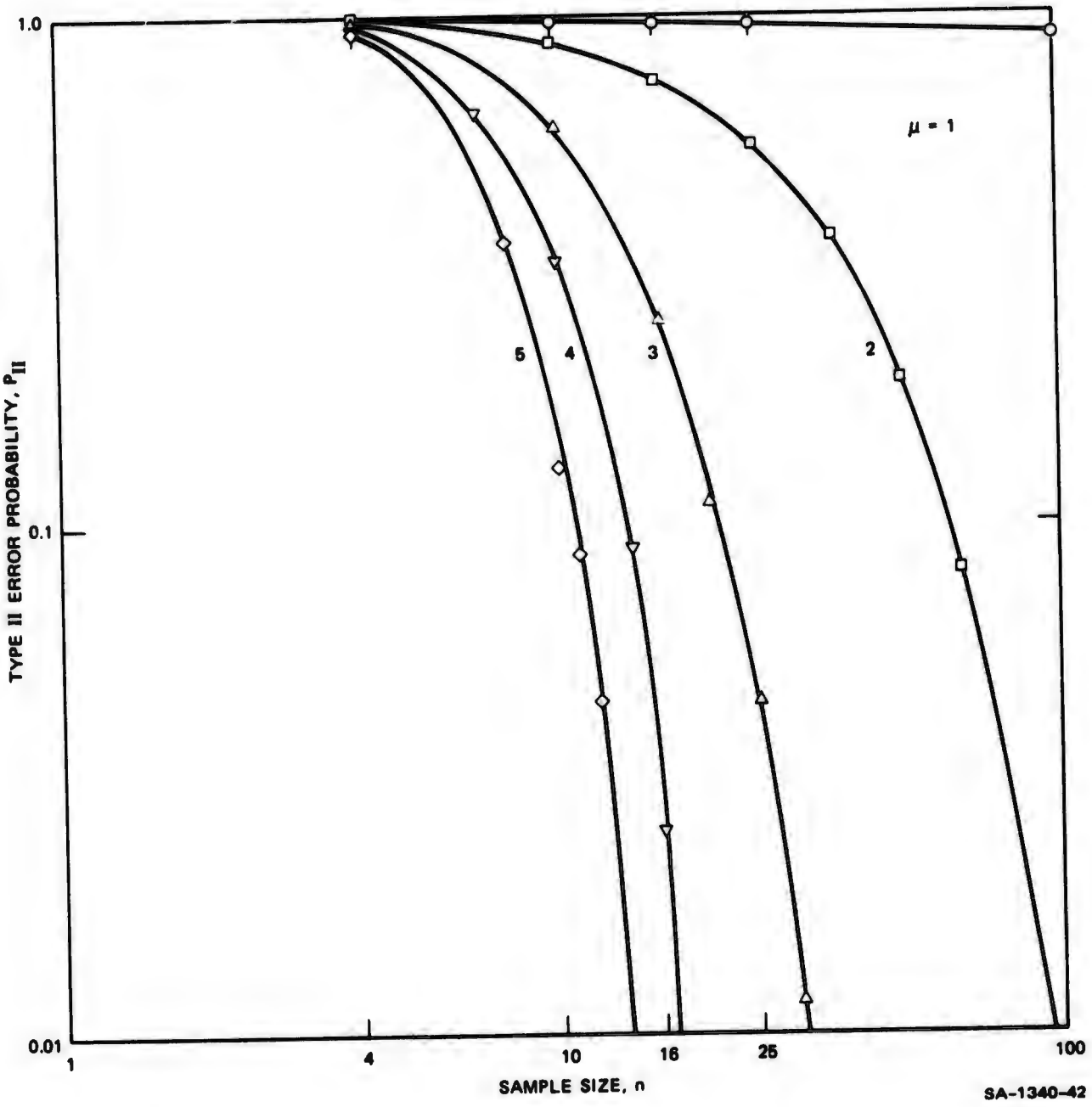


Figure 22. Type II Error Probability for $z_{0.05} = 0.87$

The results presented in this report allow the user to determine critical values for the KS statistic for a wide range of values of Type I error rate and sample size n, and for values of dimensionality d from 1 to 5. Unfortunately, the required computation grows as n^{d+1}. This exponential growth is a very serious drawback, limiting the use of the KS test to low dimensional data.

As mentioned in Section IV-B-3, the basic reason for this exponential growth is that the maximum value of $|F_n(x) - F(x)|$ usually does not occur at one of the n sample points. To get the true maximum, one must exhaustively consider all n^d points that can be formed by selecting from the components of the n sample points. Hoping that we could find something quite close to the maximum by randomly sampling these n^d points (with replacement, for simplicity), we ran a series of experiments that used that strategy. However, the results were not encouraging. In most cases, the number of points sampled had to be essentially all n^d before the results were at all close to the exact results.

There are several possible unexplored ways in which the computational problem might be attacked. One is to partition the data into a moderate number n_c of clusters, find the data points nearest these cluster centers, and restrict the search to the resulting n_c^d points. Another is to reduce the dimensionality by any of several procedures for dimensionality reduction and to apply the KS test in this reduced space. An extreme of this approach would be to use d separate univariate tests, one for each coordinate.

Finally, other tests of goodness of fit are available and deserve investigation. The Cramér-von Mieses test is particularly attractive, since it is an integrated comparison of F_n and F that might be less sensitive to approximate evaluation. In addition, some theoretical results are available for the case where some parameters of F must be estimated

from the sample (Darling [96]). The χ^2 test suffers from the need to partition the observation space into cells, but it is possible that a clustering procedure could be used to produce an effective partitioning. If so, it has the advantage that the effect of parameter estimation can be accounted for merely by reducing the number of degrees of freedom. Finally, with some sacrifice in power, it may be possible to devise computationally simpler test statistics (such as multivariate analogs of skewness and kurtosis) that discriminate between single and multiple clusters. Bonner's test [88] and the test suggested by Duda and Hart [25] fall in this category and may represent the only computationally feasible alternatives for high dimensional data.

C. Approximate Calculation of Minimal Spanning Trees

1. Exact Algorithms

Several algorithms have been proposed for computing minimal spanning trees, including those of Kruskal [97], Prim [98], and Dijkstra [99]. The following procedure, described as a FORTRAN program by Whitney [100], is typical. It constructs the tree for a set $\chi = \{x_1,\ldots,x_n\}$ of n points in n-1 iterations, the kth iteration requiring the computation of n-k distances. If we let $d_{min}(i)$ be an array for storing the shortest distance from x_i to the partial tree at the kth iteration, and $i_{min}(i)$ be an array indicating the index for the point in that tree nearest x_i, then we can describe the algorithm as follows:

(1) Set $\chi \leftarrow \chi - \{x_n\}$;
set $d_{min}(i) \leftarrow$ very large number, $i = 1,\ldots,n-1$;
set $p \leftarrow 1$.

Loop: (2) For all $x_i \in \chi$, if $\|x_i - x_p\| < d_{min}(i)$,
then set $d_{min}(i) \leftarrow \|x_i - x_p\|$
and $i_{min}(i) \leftarrow p$.

This suggests doing calculations to reject large groups of points without looking at the individual members of each group. Suppose, for example, that the n points had been partitioned into g groups, $\chi_1, \ldots, \chi_g$, where Group χ_i contains n_i points. Suppose further that we represent the points in χ_i by a single group "center" m_i. Then we might attempt to find the point nearest a given point x by measuring the g distances $\|x - m_i\|$, finding the nearest center (say m_p), and measuring the n_p distances from x to points in χ_p.[4] If we assume for simplicity that $n_1 = n_2 = \ldots = n_g = n/g$, this requires the computation of $g + (n/g)$ distances. Ignoring the requirements for integral solutions, we obtain a minimum of $2n^{1/2}$ distance calculations if we use $g = n^{1/2}$ groups of $n^{1/2}$ points each. Thus, even if we repeat this process n times, the computation grows as $2nn^{1/2}$ rather than as n^2, which is an appreciable saving for large n.

Even greater savings can be obtained by forming groups within the groups. For example, suppose that each of the g groups of n/g points is itself divided into h groups of (n/g)/h points. Then finding the point nearest x would require the computation of $g + h + (n/gh)$ distances, which has a minimum value of $3n^{1/3}$ when $g = h = n^{1/3}$. Thus, this two-level hierarchical grouping reduces the computation growth for n points to $3nn^{1/3}$. A similar analysis shows that for (k-1)-level hierarchical grouping, the computation grows as $knn^{1/k}$, which has a minimum level of en log n with the optimal choice $k = \log n$.

Table I gives numerical values of the number of distance calculations for typical values of n and k. Clearly, the greatest percentage

---

[4] If the centers are not good representatives of all of the points in their groups, the nearest point may not be in the group having the nearest center. The price for greater computational efficiency is often loss of optimality.

TABLE I. THE $knn^{1/k}$ DISTANCE CALCULATIONS FOR A (k-1)-LEVEL PROCEDURE TO FIND THE NEAREST OF n POINTS n TIMES

n	1	2	3	4	log n
10	100	63	65	71	63
100	10,000	2,000	1,400	1,260	1,250
1,000	1,000,000	63,000	30,000	23,000	18,000
10,000	100,000,000	2,000,000	650,000	400,000	250,000

improvement is achieved by going from no grouping (k = 1) to one-level grouping (k = 2). For the values of n normally encountered in cluster analysis, the payoff for using higher level groupings is relatively small. Thus, in investigating approximate procedures for finding minimal spanning trees, we restricted our attention to one-level groupings.

3. The Approximate Procedure

After considering several ways of using grouping to obtain a more efficient, approximate algorithm for generating minimal spanning trees, we finally implemented the procedure described in this section. The basic idea is to form $g \approx n^{1/2}$ groups sequentially during one pass through the n data points. Each group is a (hopefully) compact cluster containing roughly $n^{1/2}$ points, although this is subject to variation. After the groups are formed, exact minimal spanning trees are computed for each group, and the trees for the groups are linked together.

Consider first the formation of the groups. The points are considered one at a time in the sequence $x_1, x_2, \ldots, x_k, \ldots x_n$. Initially, x_1 and x_2 are used as the group centers m_1 and m_2, respectively, and at the time that x_k is considered approximately $k^{1/2}$ group centers have been defined. These are updated by assigning x_k to the group with the nearest center, and by adjusting that center so that it remains the mean of the group.

When $k^{1/2}$ exceeds the number of groups, a new group is formed by splitting the largest of the old groups. The split is made by passing a hyperplane through the group center. The normal vector for this hyperplane points from the group center to the point in the group furthest from the center. Although more elaborate procedures were investigated, this simple procedure was usually found to produce two reasonable groups of roughly equal size.

After all n points are considered, an exact minimal spanning tree is found for each group.[5] In addition, an exact minimal spanning tree is found for the g centers. This latter tree is used to determine which groups should be linked. Suppose that groups χ_i and χ_j are to be joined, and that the number n_i of points in χ_i equals or exceeds the number n_j of points in χ_j. Then the groups are joined by finding the point x_p in χ_i nearest to the center for χ_j, and then finding the point x_q in χ_j nearest to x_p; an edge from x_p to x_q is used to join the trees for these two groups. The reason for starting with the larger group is that it is often less well represented by its center; if desired, an exhaustive computation could be used to find the shortest edge from χ_i to χ_j without losing the $nn^{1/2}$ computational growth.

A more precise statement of this approximate procedure is given below.

---

[5] If the number of points in a group is large, it might pay to use the same procedure recursively to find an approximate minimal tree for each group. However, since the time required for the preceding group formation process already grows as $nn^{1/2}$, greater efficiency at this point cannot improve the $nn^{1/2}$ growth rate.

Initialize: (1) Set $g \leftarrow 2$; $\chi_1 \leftarrow \{x_1\}$; $\chi_2 \leftarrow \{x_2\}$;
$m_i \leftarrow x_i$, $i = 1,2$;
$n_i \leftarrow 1$, $i = 1,2$;
$k \leftarrow 2$.

Assign: (2) Set $k \leftarrow k + 1$;
if $k > n$, go to MST.

(3) Measure $\|x_k - m_i\|$ for $i = 1,\ldots,g$, and let m_p be the nearest center.

(4) Set $\chi_p \leftarrow \chi_p \cup \{x_k\}$;
$n_p \leftarrow n_p + 1$;
$m_p \leftarrow m_p + (x_k - m_p)/n_p$.

(5) If $k \leq g^2$, go to Assign.

Split: (6) Let $n_p = \max(n_1,\ldots,n_g)$;
let x_q be the point in χ_p for which $\|x_q - m_p\|$ is maximum;
let $w = x_q - m_p$;
set $g \leftarrow g+1$.

(7) For all $x_i \in \chi_p$, set $\chi_p \leftarrow \{x_i \mid w\cdot(x_i - m_p) > 0\}$;
$\chi_g \leftarrow \{x_i \mid w\cdot(x_i - m_p) \leq 0\}$.

(8) Set $n_p \leftarrow$ number of points in χ_p;
$n_g \leftarrow$ number of points in χ_g;
go to Assign.

MST: (9) Find MSTs for χ_i, $i = 1,\ldots, g$.

(10) Find MST for $\{m_1,\ldots,m_g\}$;
let the kth edge in this tree join $m_{i(k)}$ to $m_{j(k)}$.

(11) Set $k \leftarrow 1$.

Link: (12) Set $i \leftarrow i(k)$; $j \leftarrow j(k)$;
if $n_i < n_j$, interchange i and j.

(13) Let x_p be the point in χ_i nearest to m_j, and let x_q be the point in χ_j nearest to x_p;
link the MSTs for χ_i and χ_j by an edge from x_p to x_q.

(14) Set $k \leftarrow k+1$;
if $k < g$, go to Link .

4. Computational Requirements

An exact analysis of the computational requirements of this procedure is difficult, since the results depend on the detailed nature of the set χ of n points. A rough analysis can be made by dividing the procedure into five parts:

- Assignment of points to groups [Steps (2) through (5)]
- Splitting of groups [Steps (6) through (8)]
- Computation of the MSTs for the groups [Step (9)]
- Computation of the MST for the group centers [Step (10)]
- Linking of the groups [Steps (11) through (14)].

During point assignment, approximately $k^{1/2}$ distances must be computed for each k, requiring

$$\sum_{k=3}^{n} k^{1/2} \sim nn^{1/2}$$

computations. Splitting occurs when $k = j^2+1$ for j from 2 to approximately $n^{1/2}$. Since approximately j distances must be computed each time, this computation grows only linearly with n:

$$\sum_{j=2}^{n^{1/2}} j \sim n$$

Since an exact procedure is used to compute the MSTs for the $n^{1/2}$ groups, and since each group contains roughly $n^{1/2}$ points, the computation for this part grows as

$$n^{1/2} (n^{1/2})^2 = nn^{1/2}$$

The computation of the MST for the group centers involves only $n^{1/2}$ points and thus grows only linearly with n. Finally, the linking of the groups requires $n^{1/2}$-1 computations of roughly $2n^{1/2}$ distances, $n^{1/2}$ from points in χ_i to the center m_j, and $n^{1/2}$ from the point x_p to points in χ_j. Thus, this computation also grows only linearly with n. It follows that the assignment of points to groups and the computation of MSTs for the groups dominate the computational requirements for large n, leading to a $nn^{1/2}$ growth rate.

These conclusions were checked experimentally by repeatedly executing the subroutine for the approximate minimal spanning tree for various values of n from 10 to 1000. When points in one dimension [6] were used and the program was executed on a PDP 10 computer, the results shown in Figure 23 were obtained. These results show that the execution time T_a is given (in milliseconds) by

$$T_a = 1.31\ n^{1.33}$$

---

[6] Since the time required for distance calculations is proportional to dimensionality, the computation times for d-dimensional problems can be estimated by multiplying the one-dimensional results by d.

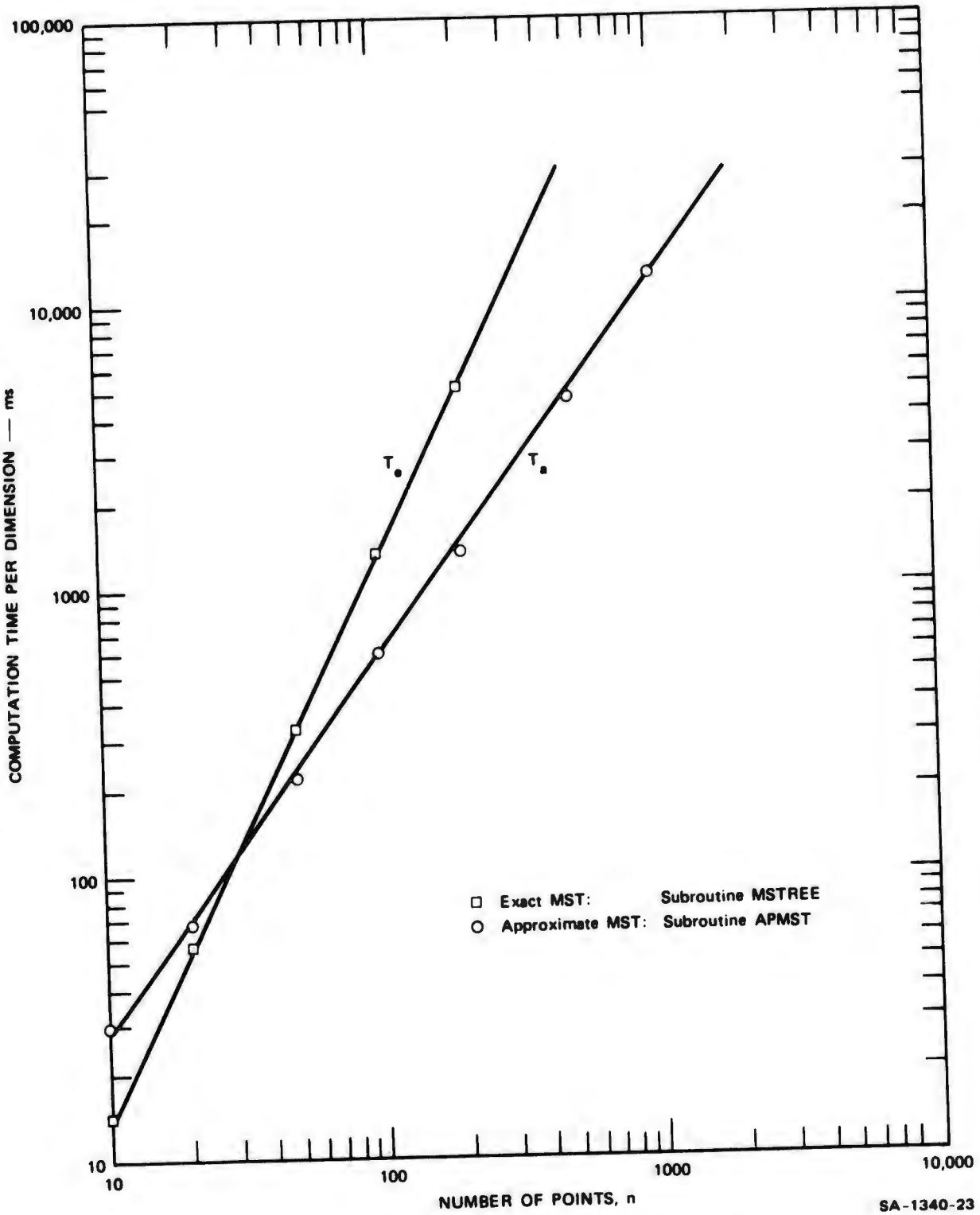


Figure 23. Computational Requirements for the Minimal Spanning Tree

Thus, the experimentally determined growth rate was even less than $n^{1.5}$, at least for the values of n investigated. For comparison, the exact procedure used as part of the approximate procedure was tested in the same way. The results, also shown in Figure 23, confirm a quadratic growth rate:

$$T_e = 0.137\ n^{2.00}$$

The crossover occurs for n = 32 points; for fewer than 32 points, the exact procedure is faster, whereas for more than 32 points, the approximate procedure is faster. For n = 1000 points, the approximate procedure is more than 100 times as fast as the exact procedure, and the ratio continues to increase as n increases.

5. Evaluation of Results

Since the approximate procedure is not guaranteed to produce a minimal spanning tree, an important question concerns how well the resulting spanning tree approximates a minimal spanning tree. This section describes several experiments that were performed to provide both quantitive and qualitative answers to this question.

The first series of experiments used samples from a bivariate normal population, $p(x) \sim N(0,I)$. Figures 24, 25, and 26 show typical results for samples of size 50, 200, and 500, respectively. From visual appearance, the minimal and the approximate minimal spanning tree appear to be generally similar, although closer examination discloses numerous minor differences. This is particularly noticeable in Figure 25, where the convex hulls of two groups overlap, and the approximate tree appears to contain a closed loop. If desired, this defect could be eliminated by reforming the groups just prior to finding their MSTs so that points closest to m_i are placed in χ_i.

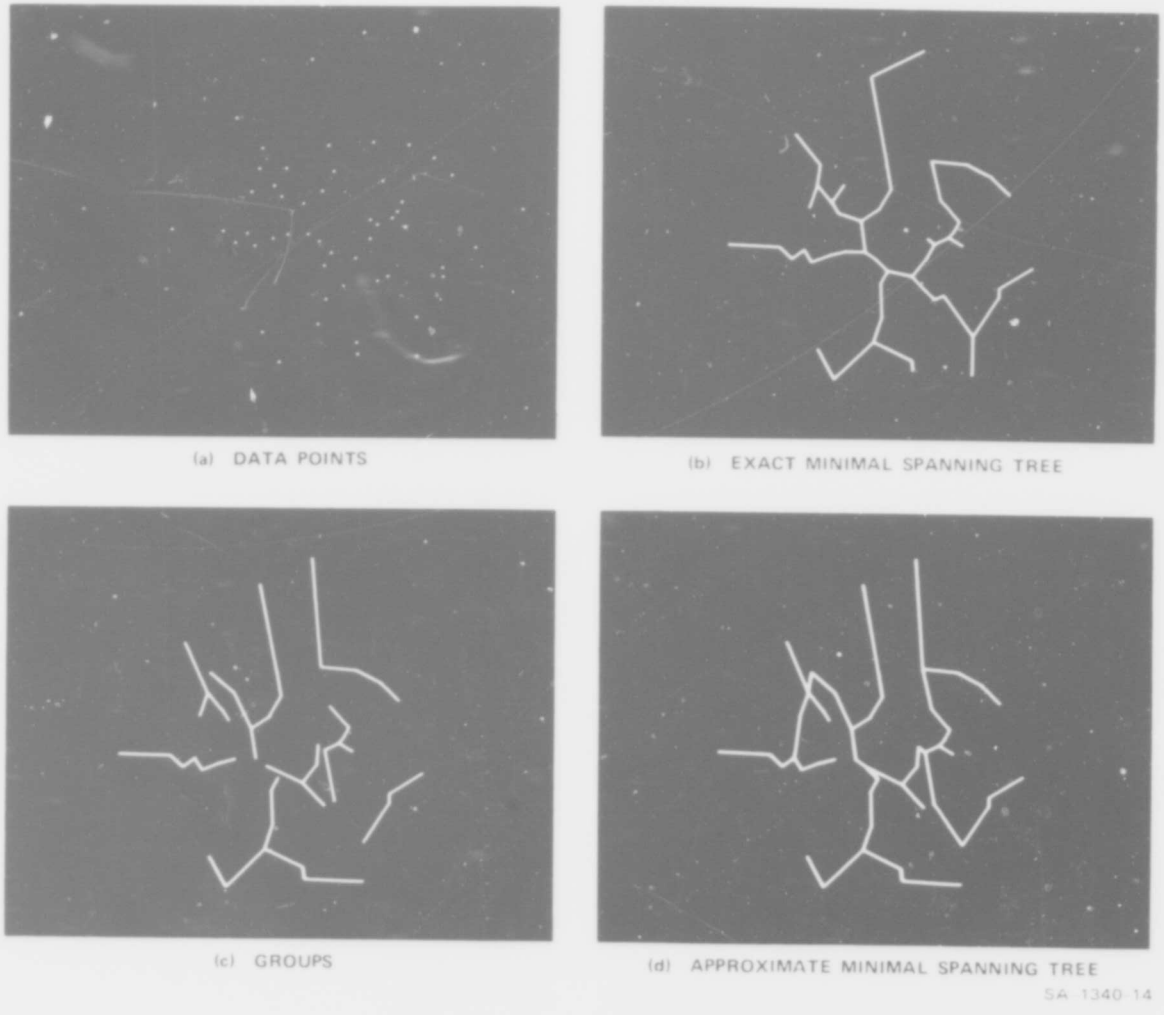


Figure 24. Spanning Trees for 50 Normally Distributed Points

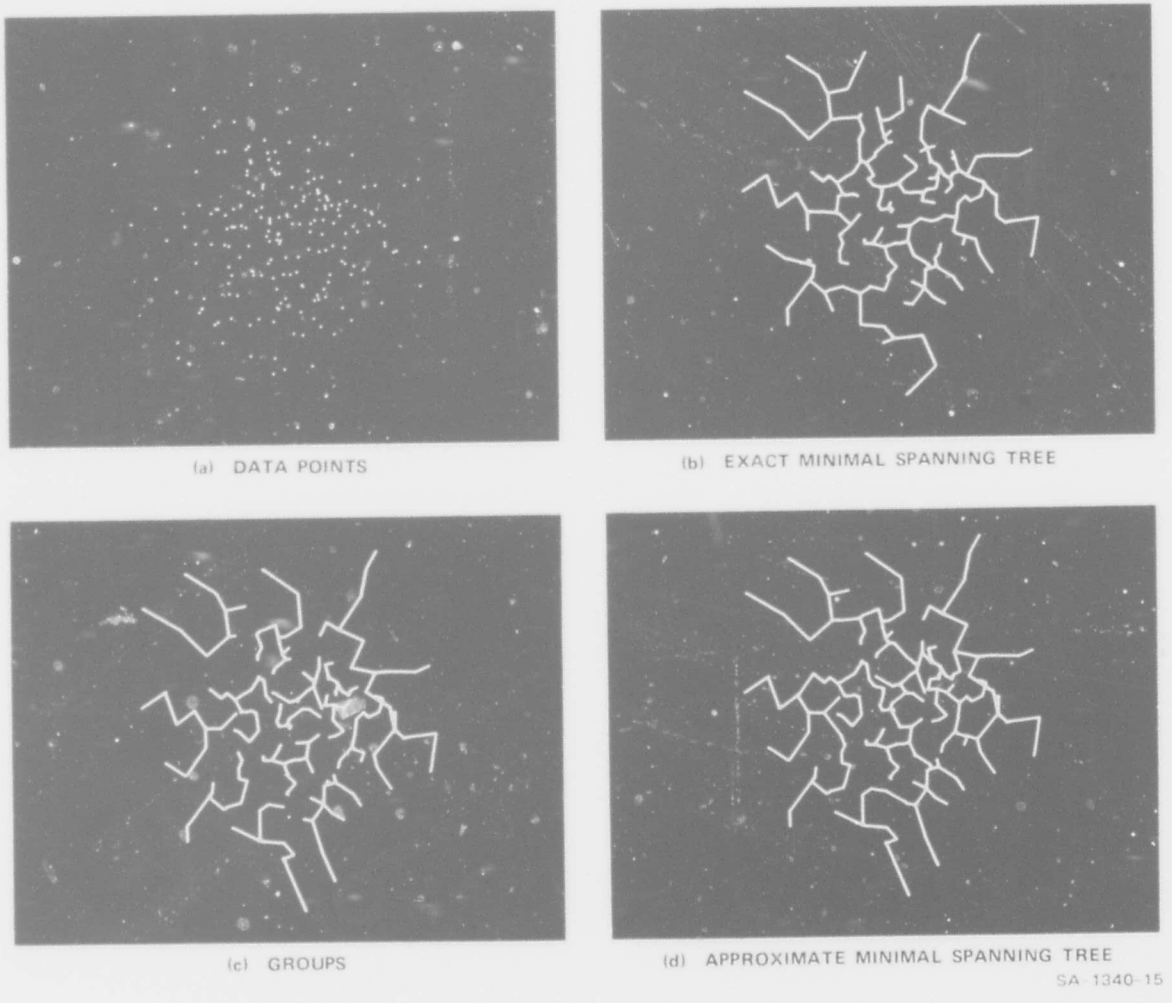


Figure 25. Spanning Trees for 200 Normally Distributed Points

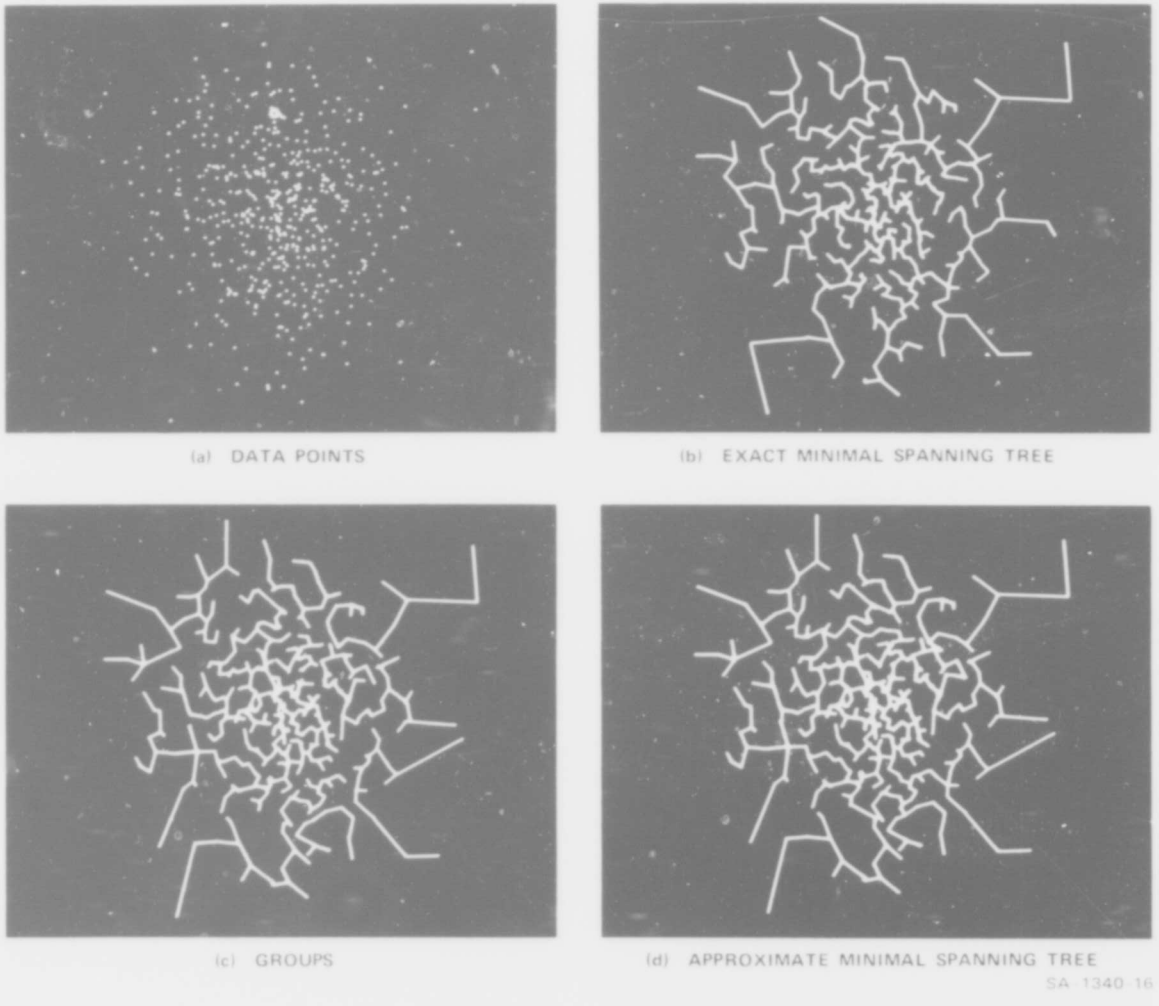


Figure 26. Spanning Trees for 500 Normally Distributed Points

To obtain a quantitative measure of how well the approximate tree approximates the minimal tree, the sum of edge lengths S was calculated for 25 bivariate normal samples of size 100. The results, which are summarized in Table II, show that the sum S_a for the approximate tree is on the average about 11 percent higher than the minimal sum S_e, the 2 σ range being from 5 to 17 percent. Less systematic observations confirmed these percentage results for other sample sizes.

TABLE II. STATISTICS FOR THE SUM OF EDGE LENGTHS FOR 25 SAMPLES OF SIZE 100. S_e is for the exact MST, and S_a for the approximate MST.

Measure	S_e	S_a	S_a/S_e
Maximum	31.22	34.68	1.161
Minimum	24.98	27.57	1.038
Mean	27.62	30.62	1.109
Standard deviation	1.57	1.89	0.030

Finally, the following examples illustrate the behavior of the approximate procedure on distinctly nonnormal data. The data in Figure 27 came from a mixture of three normal populations. The spanning trees for each component are what one would have expected for isolated normal samples, and Zahn's method [52] for detecting the connecting edges seems equally applicable to either the exact or the approximate tree.

The data in Figure 28 were an artificial combination of a normal cluster and line-like elements. The approximate tree is an unusually close approximation to the exact tree in the case shown. In contrast, Figure 29 shows a total failure of the approximate tree to reveal the interlocking spirals that are so evident to the eye. The reason for

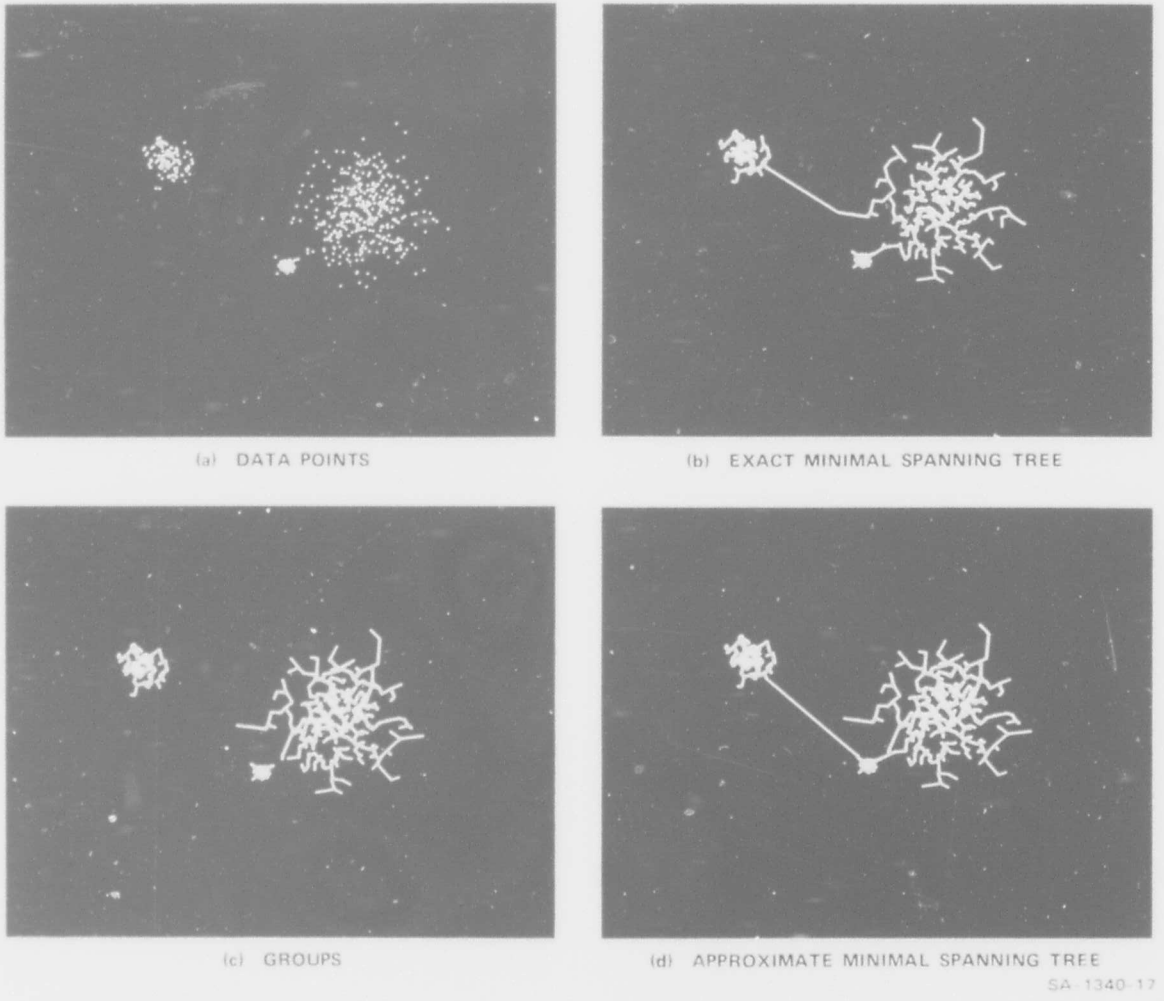


Figure 27. Spanning Trees for Points from a Normal Mixture

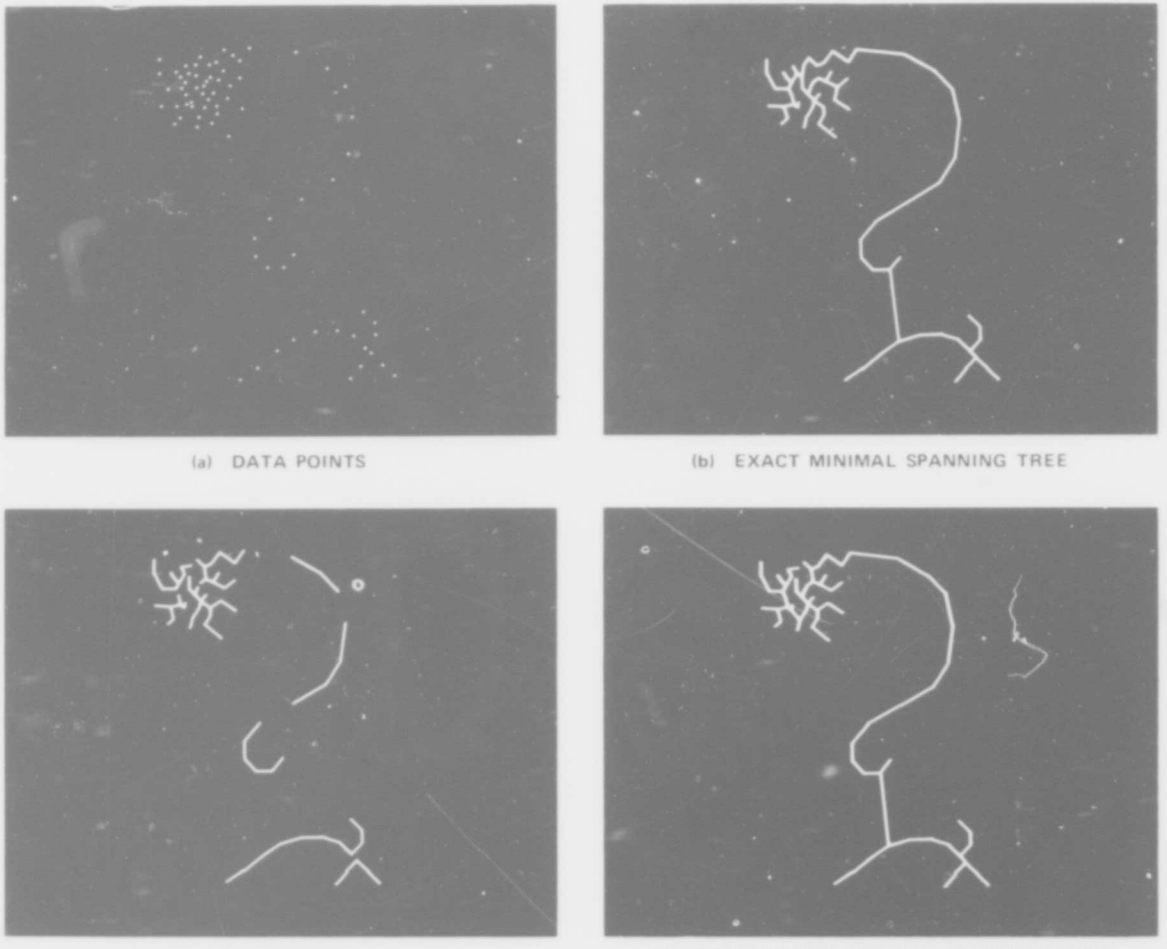


Figure 28. Spanning Trees for an Arbitrary Point Set

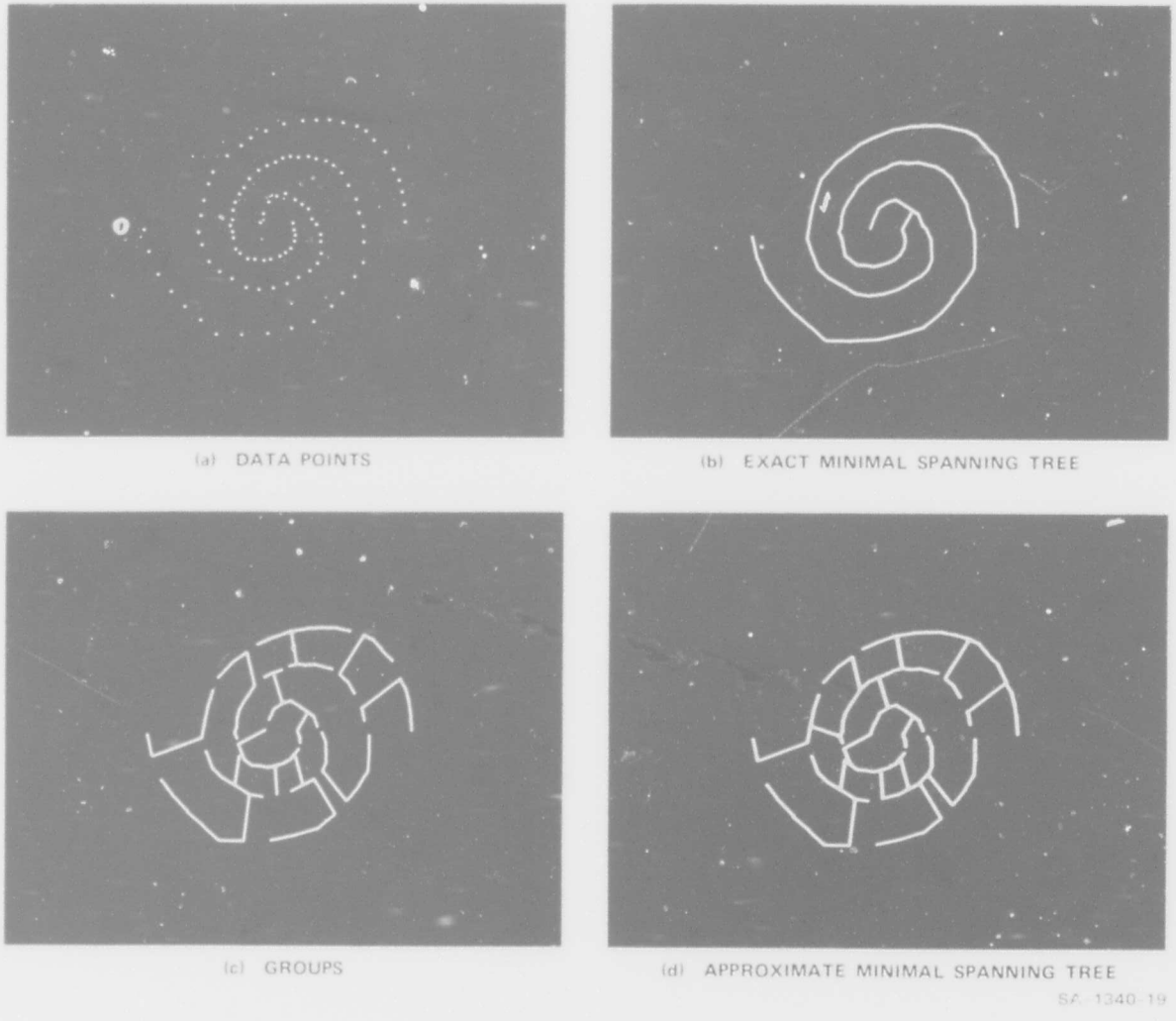


Figure 29. Spanning Trees for Interlocking Spirals

this failure is that the grouping procedure is completely isotropic, forming groups without regard to special directional structure in the data. It is possible that this defect could be remedied by a simple post-processing of the groups at the time the groups are joined--for example, by removing the longest edge in a group if it is much longer than the edge joining the groups. However, such ideas are only conjectural at this time.

SECTION V

MEASURES OF RELATIONSHIP

This section discusses fundamental concepts of distance, space, and relationship, which are the basis for clustering algorithms. We also discuss the general concept of representing many particular examples by means of a representative center, and we give several particular criteria for determining cluster centers.

A. Inadequate Measures

All clustering procedures involve, at some stage, a measure of relationship between two data points, [26], or more usually, between many data points taken in turn and a set of cluster centers. This measure (sometimes called the clustering criterion) is necessary to answer the basic question of cluster membership for each data point in the data set. This topic is also related to the problems of scaling [101-103]. We assume that each point belongs to only one cluster at a time. We now use the term "relationship," rather than "similarity," used in previous discussions [104]. (Later we consider some of the ideas of Einstein's relativity and show how they have influenced our measures of relationship.)

In the following discussion, we argue that simple Euclidean distance is not a suitable measure of relationship in general, and we give examples of its inadequacies for some simple clusterings. The need to associate each data point with an uncertainty region (or cluster domain) is also argued from basic principles of both physics and clustering. We discuss the basic notions of data description, in terms of data elements that are circular or linear. The circular or spherical model was used in our ISODATA algorithm, and the linear, string, or chain model of nodes

Preceding page blank

and links was the basis of the MST, discussed previously. We then consider the nature of the influence that points in various configurations have on each other. These questions are all fundamentally concerned with the determination of relationship between points, and with the way in which the effects of separate points can be combined to provide an overall gestalt or clustered effect. We consider some specific measures for various kinds of centers and their properties. The intent of our discussions is to show that our broad investigation of this field validates our choice of the unifying measure of relationship that we incorporate in our clustering algorithm.

1. The Inadequacies of Euclidean Distance Measures in Clustering

It is apparent from many examples that we have tested in our clustering programs [105] that the Euclidean distance measure of relationship is effective for many clustering problems [104]; however, it is easy to show by some very simple examples that Euclidean distance is inadequate except as an approximation that needs refinement.

Consider sample points for clustering as material objects in a gravitational system. For two identical clusters or points and a third point located very far from the initial two points, we can duplicate the gravitational or influential effect of the two points upon the third by replacing them with a single point that has twice the mass of the two initial points. This replacement point is centered between the original two points.

When the two initial points are not identical, the location of their combined influence shifts toward the point having the larger mass. In the extreme case where one of the points disappears, all of the influence must obviously be transferred to the remaining point. Thus,

the influence of two bodies of arbitrary mass appears to emanate from a point somewhere between the two bodies, commonly known as the center of gravity.

However, it may not be useful to consider two bodies separated in space as having a common source of influence on a third point in space because we can visually perceive two separate clusters rather than one body having a combined influence. As long as our human visual judgment perceives the clear separation of two clusters, we must accept the fact that it is not consistent with the simple concept of center of gravity.

An example of this kind of confusion of gravitational effects is that the combined center of gravity of the earth and moon, i.e., the point that has an elliptic orbit about the sun, is a point approximately one-third of the way inside the crust of the earth. Thus, the earth-moon combination exerts a gravitational influence on the sun from a point much closer to the center of the earth than to the center of the moon. Yet, an observer in space, such as an astronaut relatively close to earth compared to the sun, can clearly visually distinguish between the earth and moon. A similar effect is observed with the current clustering algorithms using Euclidean distance when experimenting with a large cluster and a small one close to it. The human views the clusters as separate; yet, Euclidean distance concepts do not separate them in the algorithm.

2. Examples of Inadequate Clusterings

Some simple data types cannot be adequately clustered or represented using the Euclidean distance relationship in our clustering

algorithm.[7] A useful presentation of such data types is given in a paper by Zahn, [52], who has brought our attention to the need adequately to cluster all types of two-dimensional data.

Figure 30 shows the results of clustering some two-dimensional data.

In Figure 30, each cluster or collection of sample points is represented by the center of gravity of those points. (This is a summarization or approximation procedure that gains brevity through reduced detail--the essence of clustering.) We partition the data samples or assign them to certain clusters according to their distances from the cluster centers. A sample is assigned to its closest cluster center, thus defining the partition boundaries between clusters as the perpendicular bisectors of the lines joining the cluster centers. Note that in each case we are considering the clustering at the two-cluster level.

In Figure 30(a), the perpendicular bisector between the cluster centers cuts off part of the larger cluster, and the center of the small cluster is pulled over toward the larger cluster. We would intuitively visually prefer the partition boundary between the clusters to be in the sparse region; but then it would not be a perpendicular bisector of the line joining the two cluster centers. (The perpendicular bisector is the

---

[7] "One is ordinarily accustomed to study geometry divorced from any relation between its concepts and experience. There are advantages in isolating that which is purely logical and independent of what is, in principle, incomplete empiricism. This is satisfactory to the pure mathematician. He is satisfied if he can deduce his theorems from axioms correctly, that is, without errors of logic. The questions as to whether Euclidean geometry is true or not does not concern him. But for our purpose it is necessary to associate the fundamental concepts of geometry with natural objects; without such an association geometry is worthless for the physicist. The physicist is concerned with the question as to whether the theorems of geometry are true or not." (Einstein [106]).

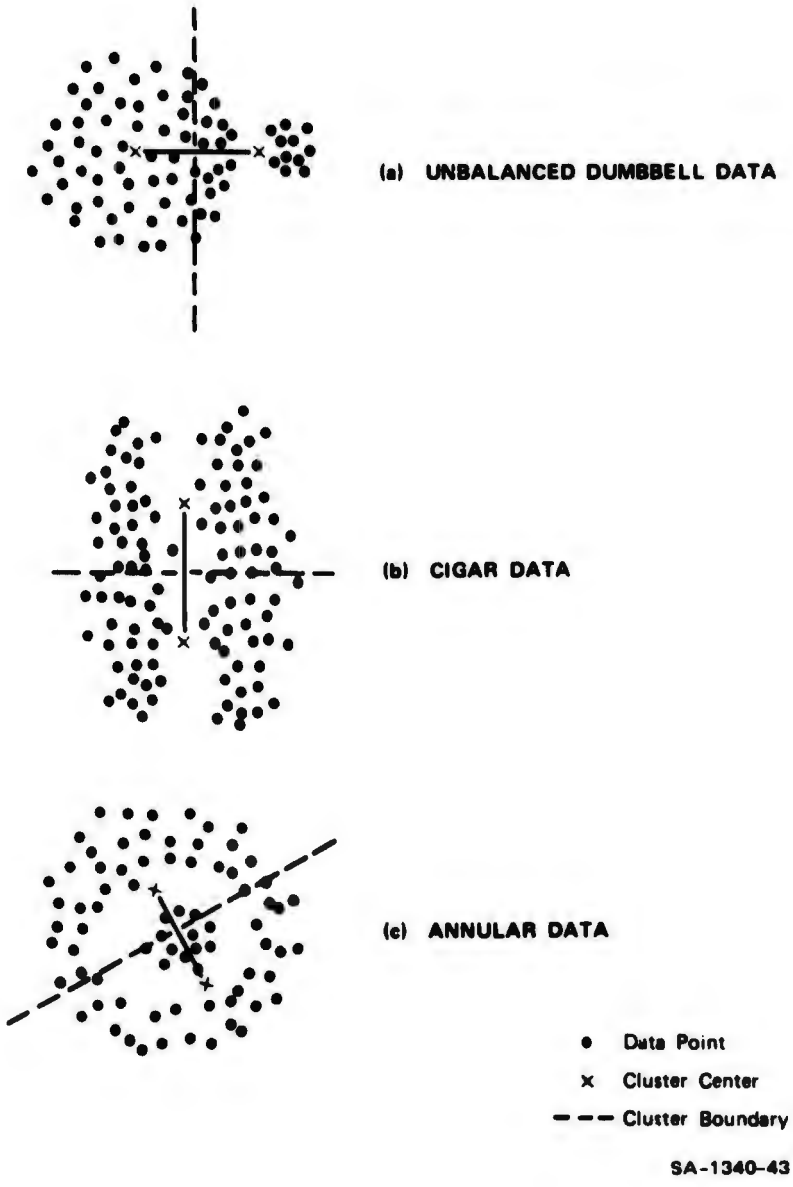


Figure 30. Three Simple Data Types That are Difficult to Cluster Adequately

partition boundary between clusters as consequence of using Euclidean distance, in the ISODATA clustering algorithm, as the cluster membership relationship measure, as sketched later in Figure 34. In Figure 30(b), we would again intuitively prefer to have the partition boundary in the sparse region between the elongated clusters.

For the annular data shown in Figure 30(c), we would intuitively prefer a circular boundary dividing the membership of the inner points from those in the outer ring. Note that in this case the cluster centers may be close to each other, but their domains of interest are different. In the cases of Figures 30(a) and 30(b), clustering is not difficult, except when the distance between the two clusters drops below a crucial range. Also, if several clusters are used, the resulting boundaries using Euclidean distance can be quite satisfactory, even for the data of Figure 30(c). We discuss these data and problems more fully later in connection with the computer experiments.

3. The Inadequacies of Dimensionless Points

The simplest data element is a single point. Each point has a position (usually in a multivariate space) and an extent. We do not accept the concept of a dimensionless point, except sometimes as a convenient fiction for some special theoretical purposes. Here we are more interested in practical and realistic applications in which each data point has a size or standard deviation.[8] Thus, we make no essential distinction between a point and a cluster or between a sample and its parent distribution because we need a unified treatment that does not make unnecessary distinctions between various levels of data types. In this way, we can iteratively apply the output (conceptual results) from one clustering (usually cluster centers) as input (usually data points) to another stage of the basic clustering process.

---

[8] For example, in our cloud clustering applications, each point in the satellite picute data represents an extent of approximately 2.5 miles on the earth.

This brings up both the awkward question of the standard deviation of one point, and the estimates of precision of measurements for the initial data or measurement points. Note also the physical absurdity of infinite precision implied by dimensionless points or elementary particles.

We treat the problem of dimensionless points by referring to the original measurement process that provided the data coordinates. We know that this measurement was not infinitely precise [107], and therefore, the position of each point is associated with an uncertainty region [108]. We loosely refer to this region as the standard deviation of each data point.

B. General Concepts for Determination of a Center

An important topic in the visual formation of clusters, and also in the formation of other psychological concepts [109], is the notion of the determination of a center. (We are using this terminology in a very broad sense.) In statistical terms we are discussing measures of central tendency. The very concept of a center implies the existence of a boundary that is most simply circular. The contents, quantities, data points, or essence of the matter within the boundary, modifies or contributes to the center concept. For example, in the case where the boundary is circular, the contents within the circle must support the circularity of the boundary; i.e., the gestalt or synthesis of parts or particles that contribute to the unifying principle (i.e., cluster center), must not contradict the hypothesis of circularity.

When the circularity is a weak hypothesis, and the distribution of particles within the body does not lead to a strictly circular boundary, we still retain the loose concept of a center. An alternative concept, perhaps the next most simple, is that the body is linear. This implies

that the cluster is extended in space without significant substance or breadth compared to its extension. A typical example of this is the fitting of a regression line or least-squares fit to approximately correlated data.

Thus, two simple and useful models for describing data structure are: (1) circularity and (2) linearity. They both have a similar order of complexity, since the circular model implies a center and a boundary, giving the concept of size or body; and the linear model gives no concept of body but does imply a length or extent. Note also that in the plane, each model requires two parameters to specify the model. For the circle, these parameters are the location of the center and the diameter or radius of the circle. For the line, they are the location of the two end points of the line.

We have made extensive use of the circular model (hyperspherical in multivariate space) in our ISODATA program for clustering; in regression methods, the linear model is used. The MST method also uses a basically linear model. In complex data sets, the choice can be made to resort either (1) to combinations or unifications of the simpler models to describe the data or (2) to construction of a less simple model (e.g., triangles, ellipses, or more transcendental functions) that more adequately describes the data. The trade-offs and cost/benefit considerations depend upon the problem, i.e., the data and the environment of application [110].

C. A Unifying Viewpoint for Clustering Problems

Certain general questions about the treatment of data points in a feature space recur which are independent of the application. For instance, into what number of subsets do the data points most naturally seem to fall? What are the extents of and boundaries between these

subsets? How compact or complex are the subsets? How is the complexity or compactness of one subset influenced by other subsets? This section attempts to find a unifying viewpoint for these questions.

In a number of different problems related to clustering a critical issue is how to compute relatedness or relationship among points within clusters and among clusters as functions of distances between the various point pairs. Different functions lead to different styles of solutions for these problems.

Whenever we attempt to put a number of data points, objects, or patterns into a smaller number of classes or categories, we must ultimately rely on being able to measure one or more of their features or characteristics. Each characteristic can be thought of as corresponding to a coordinate of a multidimensional space in which points (or vectors) correspond to objects. Two points that are in some sense close to each other should likely be put into the same category, and those pairs separated by greater distance should likely be put into different categories.

There are many different spaces in which objects can be thought to exist because there are always many different measurements and combinations of measurements that can be made. Whether or not two categories of objects are easily separated in a space is directly related to what we call the appropriateness of that space. For a given set of objects, the appropriateness of the space depends upon the purposes of the categorization. For instance, separation of positive and negative integers into positive and nonpositive categories (by comparing each with a threshold of one-half) is simple when those integers are thought of as vectors that correspond to the integers in the ordinary way. On the other hand, separation of these integers into even and odd categories by a single threshold is impossible for that space but easy in a space in which the

single coordinate is the value of the least significant digit in a base-two representation.

In the latter example, all objects in a given category are mapped into a single point of the space. This is what we might consider an ideal situation. There are exactly as many occupied points in the space as there are categories. More generally, however, objects in a given category are mapped into a cluster of points that is either compact or loose, depending on both the appropriateness of the space and the extent to which the accuracy of the measurements was affected by noise. It is convenient for us to assume in much of the discussion that follows that the clusters of points exist in an appropriate space and that the measurements are accurate enough that a cluster of points representing a given category can be surrounded by and separated from other clusters by relatively simple boundaries. (An abstract boundary that is the union of a multiplicity of subboundaries, one surrounding each point of the set, always exists.) In fact, the appropriateness of a space in which objects are represented corresponds to the simplicity of the boundaries that are then possible.

A variety of interrelated problems have clustering as their central theme. Some of these are mentioned below to put the specific results of this report into better perspective.

For a single cluster (with no attempt being made to delineate two or more categories), probably the single most important problem is that of finding the center (or the most centrally located member or the most typical point) of the cluster. Usually an acceptable solution to this problem gives us insight into the complementary problem of determining the least typical (or most peripheral or most extraneous) points of the set.

For two or more categories, the corresponding problems exist for the individual component clusters. In addition, there exists the problem of determining "natural" boundaries between pairs of clusters based on some measure of simplicity for those boundaries. Furthermore, if we are told to assume that a certain number of categories exist, then the identities of the best boundaries depend upon that assumed number. The partitioning of a point set into two categories may possibly suggest to us quite different boundaries from those that are "natural" if we are told to partition the points into three categories.

Probably the simplest form for a cluster is one with circular symmetry.[9] Even when clusters do not overlap, however, they may have forms other than circular symmetry. In that case, it is often useful to imagine that the ideal or prototypical objects are represented by some sort of curve in the space and that the points of a given cluster occur as perturbations away from those ideals. In such a case, we refer to the cluster as structured (rather than simple, a term that we apply only to clusters having a single center). For structured clusters, the problem of determining the underlying skeleton of prototypes is the generalization of the problem of determining the center of a simple cluster.

Finally, when centers or skeletons of boundaries between clusters are determined, there exists the problem of establishing some sort of compactness (or, inversely, complexity) measure for the cluster. If we assume that a cluster has a center (or skeleton) corresponding to the idealized member(s) of the cluster, complexity or compactness measures would be, respectively, positively or negatively correlated with the amount of noise associated with the measurements of the underlying

---

[9]More generally, it would have spherical symmetry. Here we use terminology appropriate to the two-dimensional situation even when it is clear that the concepts are valid for a space of arbitrary finite dimension.

objects or with the inherent variability among the objects that have been placed in that category. Those measures could, in turn, be useful in determining which boundaries between clusters are the best ones.

1. Particular Criteria for Determining Cluster Centers

Undoubtedly, the simplest criterion for finding the center of a cluster is that it be placed at the center of gravity of the point set. The center thus determined has the property that from it the (scalar) sum of the squares of the (Euclidean) distances to the other points of the set is minimum. This concept is valid even when the points of the set have differing weights. In that case, the squares of the distances should be multiplied by the weights of the respective points. Alternatively, the vector sum of the vectors (having magnitudes that are the products of the weights of the elements and their distances from the center) from this center to the various points is then zero.

Another simple definition of center is one that results in minimizing the (scalar) sum of the distances to the points of the set, multiplied by the weights of those points if all points do not have the same weight. The (vector) sum of the vectors, each of which is directed from that center to a point of the set and which has a magnitude equal to the weight of that point, is zero. Such a center can be thought of as requiring the least work in carrying the weighted points of the set from that center to their locations in the space [111].

The above two definitions may be thought of as special cases of a more inclusive definition of center for an n-element set of points of different weights in a d-dimensional space. This more general center is defined as the minimum of a function of d variables obtained by adding together n different components, each having the same shape, each centered on a point of the set, and each having an amplitude proportional

to the weight at that point.[10] (The common shape of the components can be chosen to be one that increases monotonically with distance, as in the preceding two examples, although that is not a necessary restriction.) Thus, the function, the minimum of which is to be chosen as the desired center, can be thought of as resulting from the d-dimensional convolution of the point set with the chosen characteristic distance function. Alternatively, we can think of the function as the output of a d-dimensional linear filter that has as its input the point set. Choosing different filter (or characteristic) responses results in different definitions of center. The impulse response function reflects how the influence of one point on another depends on the distance that separates them.

It is also possible, of course, to consider the center to be at the maximum (rather than minimum) of the function resulting from the convolution. In this case, a characteristic function that decreases (rather than increases) with distance is most convenient.

The convolution/filter viewpoint is perhaps most natural when the characteristic function is one that is symmetric in all d variables; i.e., one in which the relatedness of two points is a function only of the distance that separates them, rather than of the direction from one to the other. There is, however, no need to limit the function in this way or even to insist that it be monotone. A spatial filter having a characteristic response matched to some particular type of pattern can be utilized to give an output that is especially high at those locations in the space where that kind of pattern exists.

---

[10]This is essentially the Parzen window procedure [112].

2. Special Cases of Interest

The two earlier examples, center of gravity and minimum work, correspond to filters that have characteristic functions that increase as the square of distance or linearly with distance, respectively. Other special cases show the power of the convolution/filter viewpoint. For instance, a characteristic response that increases arbitrarily rapidly with distance was found in the limit to yield a minimum--and therefore to identify a center for the cluster--at the center of the smallest circle within which the point set can be contained. An example of such a center is shown in Figure 31.

If, at the other extreme, the characteristic function is one that decreases arbitrarily rapidly with distance, then in the limit the function resulting from the convolution has its maximum--and thus defines a cluster center--at one of the points of the set having the minimum possible distance to its nearest neighbor. Except for pathological cases, this limits the candidates to some pair of points of the set. To determine which of the candidate points is the center, we next consider the second-nearest neighbors of the candidate points, eliminating those candidates that do not have the minimum possible distance to these second-nearest neighbors. If more than one candidate remains, the distances to the third-nearest neighbors are considered, and so forth. When all distances between pairs of points are different, as in the example shown in Figure 32, the center is the point of the pair of points separated by the least distance, which has the lesser distance to its next-nearest neighbor. In our example, the distances have been labeled so that $d_1 < d_2 < d_3 < \ldots < d_{10}$. Therefore, the center is at the point labeled E.

There are an infinite number of possible characteristic functions and, consequently, an infinite number of definitions of center.

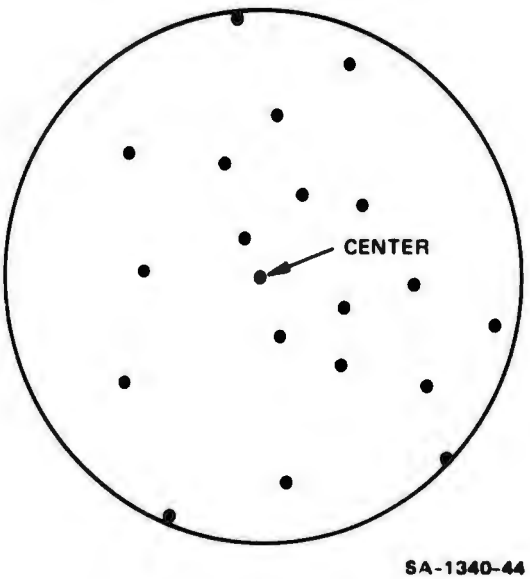


Figure 31. Cluster Center Determined by Smallest Circle Containing Point Set

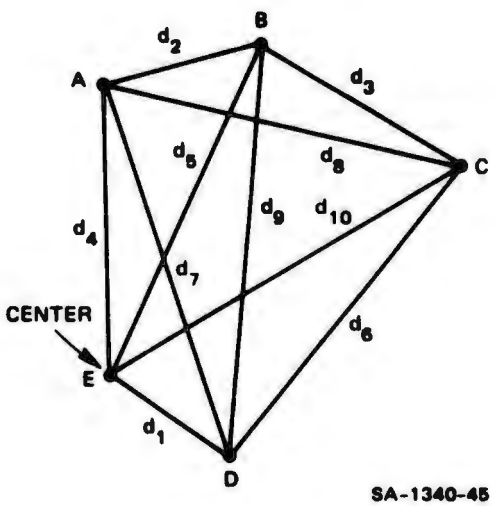


Figure 32. Cluster Center Determined by Point of Set with Minimum Possible Distance to Nearest Neighbor

For any of those characteristic functions, we might define a corresponding center, either at the maximum or minimum of the function resulting from the convolution. The most atypical or peripheral point could then be considered to be that point at which the function was a minimum or maximum, respectively.

Alternatively, we might consider the center(s) of a cluster of points to be the last point (or pair of points) to survive after the most peripheral point of the set has been deleted, the most peripheral point of the reduced set deleted, and so forth.

This latter method suggests another method (not so directly related to the convolution/filter viewpoint). The convex hull of a point set certainly contains the least central points of the set, as shown in Figure 33. When the points of the set that are on this convex hull are removed, a smaller set results. It also has a convex hull, which can also be removed. When the sequence of convex hulls is

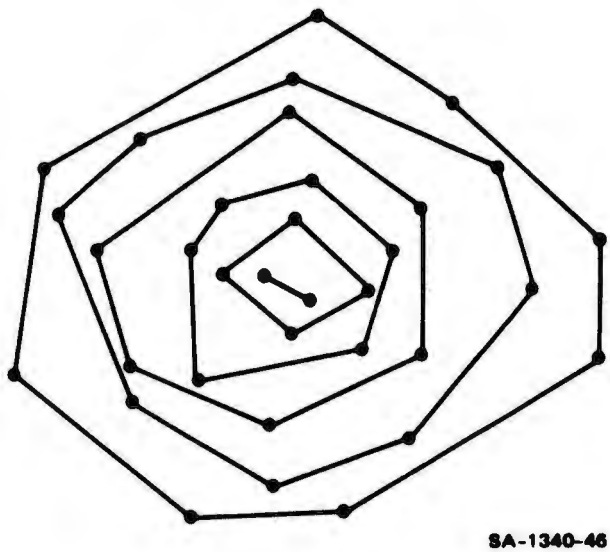


Figure 33. Bicenter Determined by Convex Hull Removal

removed, the end result is either a single point (the center) or a pair of points (the bicenter) or three or more points that constitute the inner shell of the point set.

3. Boundaries Between Clusters and Skeletons for Complex Clusters

The convolution/filter viewpoint also provides a unifying viewpoint for problems of partitioning points sets into subsets (i.e., into categories) and problems of determining boundaries for and between these subsets. As we commented earlier, the choice of a characteristic function corresponds to a decision about how the influence of one point on another is assumed to depend on the distance between them. The function that results from convolving the point set with this characteristic function (or impulse response, if we are thinking in terms of linear filtering) has a value that depends upon where we are in the d-dimensional space containing the point set. The contours that result from setting this function to constant values can be useful in determining natural boundaries between subsets of those points.

In the two-dimensional case, we can think of the function as hilly terrain and the contours as the shorelines that occur when we flood this terrain with water to fixed levels. (Here we assume that the characteristic function decreases with distance.) When the water level is sufficiently low, all of the terrain of interest will form a single island. As the water level is raised, an increased number of smaller islands result. When the water level is high and when the characteristic function decreases with distance with sufficient rapidity, there can be as many small islands as there are points of the set. Finding a natural

partitioning into a fixed number of subsets then corresponds to lowering the water level until the number of islands is reduced to the required number.

Finding natural boundaries between the subsets can also be related to viewing the function as hilly terrain. The bottoms of valleys are acceptable candidates for boundaries between hills or subsets of hills. (More formally, a point is at the bottom of a valley when it is at a minimum in the direction of one of the principal directions of curvature.)

Assume now that the characteristic function with which the point set is convolved is one that decreases very rapidly with distance. This is equivalent to saying that the value of the resulting function essentially depends only upon the distance(s) to the nearest point(s) of the set. In that case (and when all points of the set have equal weight), the bottoms of the valleys occur on lines that are the loci of points that are equidistant from the nearest pair of points in the set. These loci, shown in the example of Figure 34, correspond exactly to the boundaries between ISODATA cluster centers, if we regard each point here to represent a cluster center and assume Euclidean distance as the measure.

Note that certain pairs of points do not generate a corresponding line. This occurs when still another point of the set exists somewhere between this pair. More precisely, a pair of points of the set will generate a line only when it is possible to place these two points on the circumference of at least one circle that does not contain any other points of the set. The lines (in higher dimensional spaces, planes, or hyperplanes) partition the space into cells, one for each point of the set.

Note also that when we join pairs of points in the set that generate a line, there results a natural partitioning of the area inside

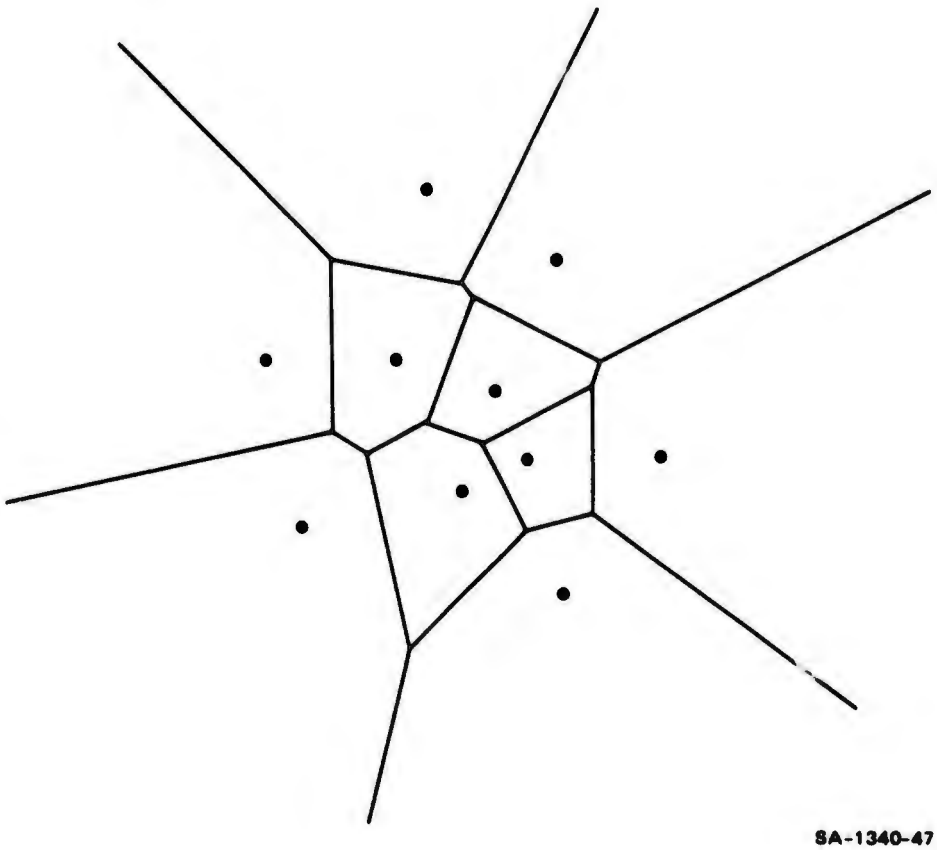


Figure 34. Decomposition of Space into Cells Associated with the Points of a Set

the convex hull of the set into triangles. Each triangle corresponds in the dual cell structure to a point at which three lines are incident. Each of these triangles corresponds to a triple of points of the set that can be placed on the periphery of a circle that contains no other points of the set.

Now associate with each line of this cell structure a value that is the distance between the pair of points that generated that line. The line having the largest value represents the valley that would be the first to flood completely as the water level is raised. That valley having the least value would be the last to be flooded, and consequently, it is associated with the pair of points closest to each other.

The concept of a minimum spanning tree may now be examined from the vantage point of this terrain model. With the water level initially high enough that it completely fills all the valleys, we then lower the level until a portion of some valley is above this level. When this occurs, an island is created that contains the two points associated with this valley. These two points are exactly those that would be the first to be joined in the procedure by which the minimum spanning tree is determined. As the water level is lowered, other valleys become evident and the corresponding points are joined by an edge in the growing tree. (When a valley becomes uncovered and the two associated hills are already on the same island, the corresponding edge is not to be added to the tree.)

When other characteristic functions are used or when the points in the set have unequal weights, the valley lines will not necessarily be straight. For each characteristic function, however, the preceding analogy can be thought of as providing a general viewpoint that leads to a satisfying definition of center (the highest point of the terrain) and a natural partitioning of the points of the set into clusters corresponding to islands.

The skeleton for a subset of points that is structured (corresponding to an island with more than one maximum) can be related to a concept complementary to that of the valley lines. We define the ridge lines for the function resulting from the convolution as the locus of points for which there is a maximum in one of the principal directions of curvature. The network of ridge lines on a given island that are entirely above a given threshold may be considered as the skeleton of the associated subset of points. For simple (unstructured) subsets, the ridge lines degenerate into single maxima and the skeleton into a single center.

4. A Natural Definition for Complexity Measure

We think of the complexity and the compactness (or simplicity) of a given subset of points (corresponding to a category of the objects associated with those points) as complementary concepts; i.e., we think of either one as some monotonically decreasing function of the other. It is most convenient to speak here of complexity measures.

For the sake of argument, assume that the cluster of points results from perturbations away from a single ideal (representing a prototype object) and that the points are represented in some appropriate space so that a cluster center can most reasonably be associated with monotonically increasing (or decreasing) characteristic functions for the filtering. Recall our earlier pair of examples: the center of gravity and least work examples. The center for the former minimized the (weighted) sums of the squares of the distances to the points; the center for the latter minimized the (weighted) sum of the distances themselves. The most natural complexity measures for these examples are obviously those same minimum sums (which could be normalized, for example, by dividing by the sum of the weights of the points).

As in these last two examples, when the characteristic function depends upon some power of the distance, the complexity measure is simply the corresponding moment of the point set about the center that minimizes that moment. By extension, for characteristic functions that are zero at the origin and that increase monotonically with distance, the complexity measure is again the minimum value of the result of convolving that function with the point set. More generally, we can think of that measure as the difference between the minimum for the actual point set and the minimum that would result if all the points had been at the location of a single prototype.

Similarly, for characteristic functions that decrease with distance (and for which the center is defined to be at the maximum of the function resulting from the convolution), the complexity can be taken to be that maximum subtracted from the maximum that would result if all points were at the same location.

When there is more than one prototype for a category of objects, the corresponding points, when plotted in some spaces, can constitute a structured cluster that may lead to a function with more than one maximum when it is convolved with a specified characteristic function. By transforming from such a space into a more appropriate space, the several original prototypes could be mapped into a single prototype. Note that it is possible to choose, for an arbitrary set of points (impulses), a characteristic function that yields a single impulse when convolved with that set of points. However, such a function would not be monotonic and, in general, would not be bounded or symmetric (depend on distance alone).

If we could determine such a prefiltering with acceptable properties, it could be applied to the original structured cluster, and the complexity measures defined earlier could be applied to the result. An ideal preprocessing would lead to a complexity measure proportional to the effects of perturbations on the original measurements on the objects to be classified. Indeed, perhaps the central problem of pattern recognition and classification is the search for representational spaces that in specific contexts of interest will require a complexity of measurements proportional to the perturbations on the original prototypes.

If we lack such a prefiltering (an ideal preprocessing of the data points may not be possible with a linear filter), or if for any reason we are forced to deal with some specified inappropriate space in which to represent the objects, we can either assume that a single center

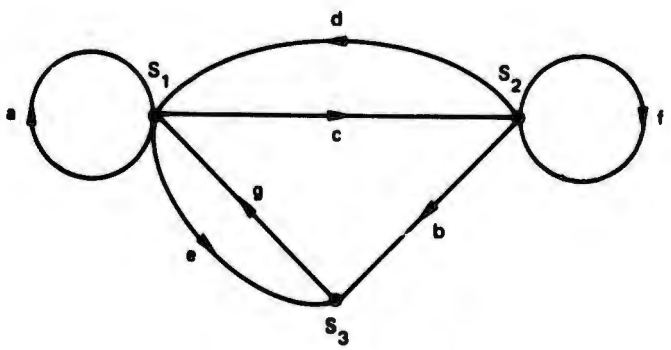
is present or we can derive a skeleton for the cluster. In the latter case, we can assume in computing the complexity that the distance to be associated with a given point of the set is its distance to the (nearest point of the) skeleton. Using this procedure, we can reduce the case of a structured cluster to the case of a simpler structure.

The next section briefly develops an alternative view of complexity related to some issues that arise in communication theory.

D. A Graph-Theoretic Measure of Compactness

1. A Problem in Information Theory

An issue of structural complexity arises in connection with a problem in communication theory. Consider the description of an information source by a state diagram, such as the illustrative one in Figure 35(a). The source is one for which an a or g symbol (both lead to State S_1) can be followed only by one of the symbols a, c, or e.



(a) STATE DIAGRAM

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

(b) INCIDENCE MATRIX

SA-1340-48

Figure 35. Representations of a Source of Symbols by State Diagram and Incidence Matrix

The diagram also shows that b, d, and f can be preceded only by c or f and that the only symbol that can follow either b or e is g. An important question for such a source is what maximum average rate of information (in bits per symbol) can be achieved. The answer to this question for a strongly connected graph is the same as that for the graph that is obtained by reversing the directions of the directed edges.

Since the maximum number of options (outward-directed edges) available at any state is three, the maximum rate can be no higher than $\log_2 3$. Since the minimum number of inward-directed edges (outward, if we consider the reversed graph) is two, the maximum rate can be no lower than $\log_2 2 = 1$; i.e., the (maximum) rate that can be achieved is at least as high as the logarithm of the minimum number of outward-directed edges at a node or the logarithm of the minimum number of inward-directed edges--whichever is larger. Furthermore the (maximum) rate can be no higher than the logarithm of the maximum number of outward-directed edges at a node or the logarithm of the maximum number of inward-directed edges--whichever is smaller

These upper and lower bounds on the maximum rate can perhaps be seen most directly by examining the rows and columns of the incidence matrix that describes the graph. In that matrix, shown in Figure 35(b), there is either a one or a zero in the ith row and jth column if there is or is not, respectively, a transition from State S_i to State S_j in the graph. Alternatively, the entry corresponds to the number of transitions from S_i to S_j.

The rate associated with a source that is described by a state diagram (which we assume is strongly connected) is a function of the conditional probabilities that are associated with the transitions. The maximum rate could be found by differentiating with respect to these probabilities (taking into account that each must have a value between

zero and one). However, the maximum rate can also be found in a much more direct way by an analysis of the graph itself or of the incidence matrix for the graph. The logarithm (base two) of the largest eigenvalue of this matrix is the desired maximum rate (also called maximum source entropy). This largest eigenvalue, x_o, can be bounded by the procedure described above.

The eigenvalues can be found by setting the determinant of the matrix M - xI (where x is a scalar and I is the identity matrix) equal to zero and solving for x. In our example,

$$|M - xI| = \begin{vmatrix} 1-x & 1 & 1 \\ 1 & 1-x & 1 \\ 1 & 0 & -x \end{vmatrix} = -x(x^2 - 2x - 1)$$

This polynomial in x has zeros at x = 0 and at $x = 1 \pm \sqrt{2}$. Consequently, the largest eigenvalue, x_o, is $1 + \sqrt{2} \cong 2.414$. The maximum source entropy or maximum rate is, therefore, $\log_2 2.414 \cong 1.25$ bits per symbol.

2. A Correspondence to the Clustering Problem

The pairs of elements of a point set bear certain relationships to each other, and a directed graph is one way of expressing those relationships. Let each point of the set correspond to a node of a graph. In that graph let there be a directed edge from the ith node to the jth node, and assign to that edge some value that in some way expresses the influence the ith point has on the jth point. The value that we choose is the product of the weight of the ith point and the value of the characteristic function (of distance) for the distance between those two points.

Since there are an infinite number of characteristic functions, an infinite number of graphs can be associated with a given point set.

The one we have studied most is $f(d) = 2^{-d}$, which seemed attractive both because $f(0) = 1$ and because $f(d_1 + d_2) = f(d_1) \cdot f(d_2)$.

The property $f(0) = 1$ implies that the graph has self-loops at each node with values equal to one. In turn, this leads to a single eigenvalue, $x_o = 1$, for an isolated point or for a set of widely separated points and a corresponding entropy for the graph equal to $\log_2 1 = 0$. In addition, this property leads to an eigenvalue $x_o = n$ for n tightly clustered points (separated from each other by distances that may be considered to be zero) and a corresponding entropy of $\log_2 n$.

Because $f(d) = 2^{-d}$, the influence that one point has on another can be thought of as resulting from a kind of attenuation that reduces that influence by a factor of two for every unit of distance. The choice of the constant two is arbitrary; a different choice is equivalent to measuring distance in different units or to a change in the overall size of a cluster. The property $f(d_1 + d_2) = f(d_1) \cdot f(d_2)$ tells us that the influence of one point on another can be considered to be the same as the influence of the first on a third (imaginary) point anywhere on the line between the first and second points, multiplied by the influence that that imaginary point would have on the second point.

An example of a point set, the corresponding directed graph, and the matrix are given in Figure 36. The physical positions of the nodes of the graph are not significant because the relevant information about distance is contained in the values associated with the edges.

Because the entropy measure tends to be large for point sets with elements that are close together, and smaller when those points constitute a looser cluster, we can take that measure to be a reasonable measure of compactness. Figure 37 shows four point sets, the determinants $|M - xI|$, and the values of the largest eigenvalues. In each of these point sets, the weights of all three elements are one and the distance

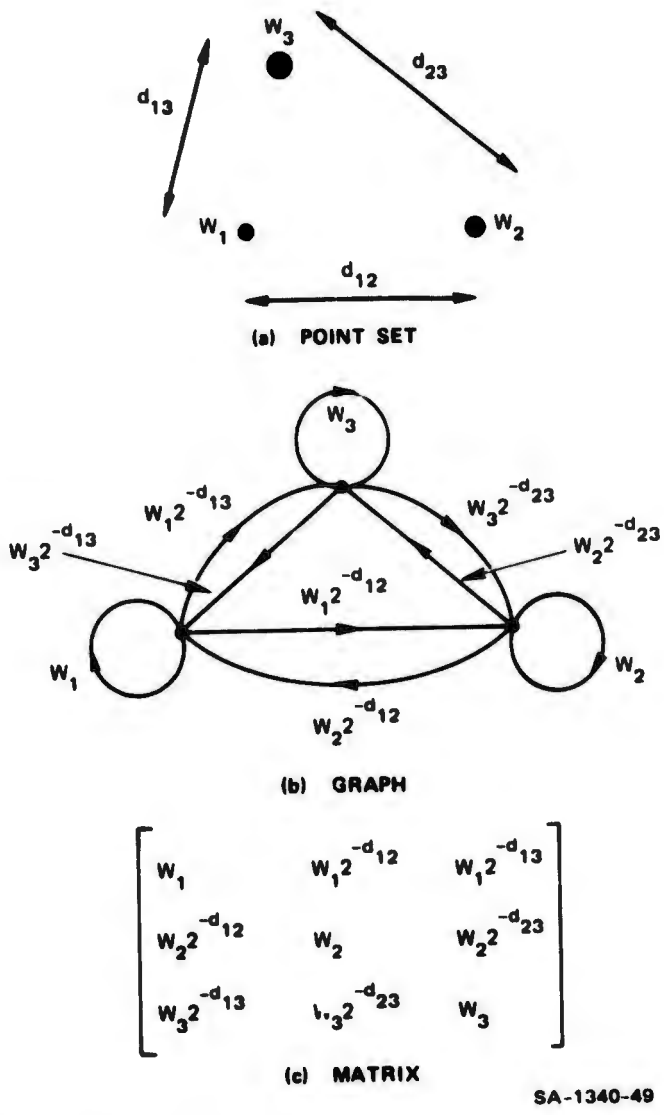


Figure 36. Sample Graph and Matrix Derived from a Point Set

separating elements is an integer (usually one). For the third and fourth clusters, the closely spaced points are assumed to have zero separation; the resulting eigenvalues are the same as if those sets were assumed to have two points of weights two and one and one point of weight three, respectively.

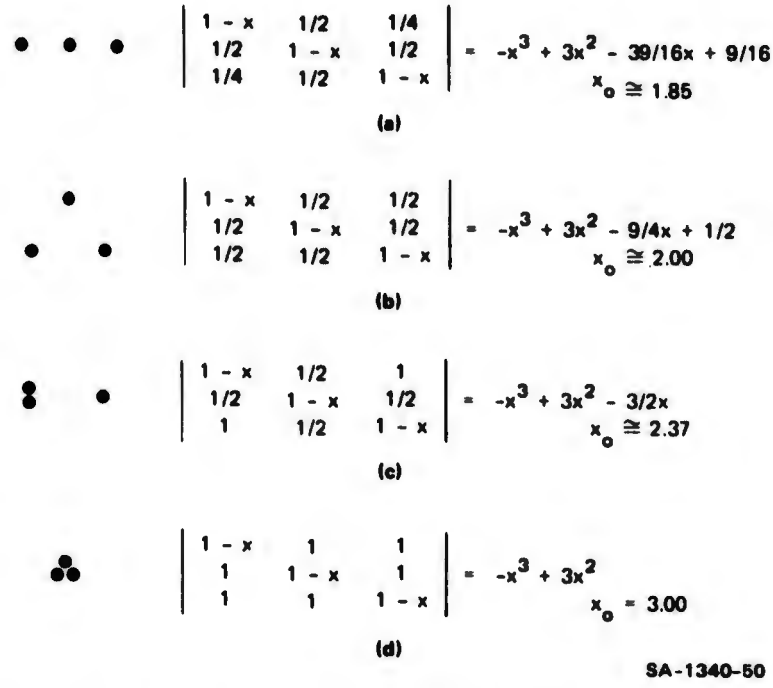


Figure 37. Four Related Point Sets with Different Measures of Compactness

Upper and lower bounds on x_o (derived by the methods described earlier) for the four examples of Figure 37 are tabulated below:

Set	Bounds
First	$1\text{-}3/4 \leq x_o \leq 2$
Second	$2 \leq x_o \leq 2$
Third	$2 \leq x_o \leq 2\text{-}1/2$
Fourth	$3 \leq x_o \leq 3$

Using these bounding methods, the entropy measure of compactness can often be estimated accurately enough (or even calculated exactly when the upper and lower bounds are the same) without having to derive the polynomial in x resulting from the determinant of the matrix M - xI. For instance, imagine an infinite set of points, each of unit weight with unit spacing between adjacent members on a straight line. A typical point has a set of distances to itself and to the other points

that is {0,1,1,2,2,3,3,4,...}. Therefore, in the (infinite) graph that corresponds to that point set, the values associated with the edges directed out of (or into) a typical node are {1, 1/2, 1/2, 1/4, 1/4, 1/8, 1/8, 1/16,...}. Thus, the largest eigenvalue is both upper and lower bounded by the sum of these values: 3. The corresponding entropy is $\log_2 3 \cong 1.585$.

Much additional work needs to be done to exploit this new measure; nevertheless, the start summarized here seems to be promising.

E. The Relevance of the Generalized Distance Relationship

We have discussed several alternative ways to determine centers and boundaries for various point set configurations, and we have noted how points influence one another in these perceptual data fields [113]. We have also noted the inadequacies of Euclidean distance measures [114] for individual perceptual phenomena in clustering. In addition, we postulate that we cannot regard data points as dimensionless and without extent because of the inevitable presence of measurement errors and uncertainty in measurement space that results. These fundamental considerations are analogous to the general state of affairs in physics when we compare our data points to particles having mass in a gravitational field.[11]

We maintain that it is unnecessary to apply all the mathematical details of relativistic physics to perception and clustering, and even

---

[11] "Since the gravitational field is determined by the configuration of masses and changes with it, the geometric structure of this space is also dependent on physical factors. Thus according to this theory space is--exactly as Riemann guessed--no longer absolute; its structure depends on physical influences. Physical geometry is no longer an isolated self-contained science like the geometry of Euclid" (Einstein) [115].

if we knew how, it may be inappropriate to do so. We suggest that the usefulness of relativity as a guide to p.r. research be tested in future experiments.

Before describing our experimental work, we consider the experimental facility. This consideration involves us in a discussion of modern experimental tools, in the form of man/machine interaction with computers. Then we discuss the particular system and language we have developed for this project for generation and description of perceptual data.

At present, the most reasonable approach to refinement of the distance measure seems to be a measurement of distance relative to the compactness of the cluster. This is the well-known concept of Mahalanobis [116], or generalized, distance, which takes the standard deviation or, more generally, the covariance matrix into account. Mahalanobis refers to Einstein's relativity as the basis for his formulation. The generalized distance from a point to a compact cluster is greater than the generalized distance from the same point to a loose or less dense cluster that is the same Euclidean distance away [117]. In Einstein gravitational terms, this means that the geometry of perceptual space is warped according to the masses in that space [115]. It seems necessary to introduce these relativistic concepts into clustering to explain the effects that we observe.

The introduction of the concept of warped space, implied by the generalized distance, also has significant consequences for individual perception psychology [106]. In engineering terms, we have noted the inadequacy of Euclidean distance to explain factually the perceived separation between simple cluster configurations. We do not lightly abandon the attractive and familiar notion of Euclidean distance, which has considerable social and objective value [102]. However, when it

fails to explain these simple clusterings, we must question its objective nature and see if it can be reconciled with our own individual visual experience. The use of a generalized distance relationship may be an essential refinement when we are dealing with real individual human perception. The widespread use of Euclidean distance is a valuable social agreement and is thus essential for technology in general, but Euclidean distance is inadequate for explaining human perception in clustering.

SECTION VI

EVOLUTIONARY DEVELOPMENT OF NEW METHODS BY MAN/MACHINE FACILITIES

In previous research we have benefited from using an interactive computer facility to perform specific p.r. application tasks. We also used the facility to diagnose its own operation. In this section, we discuss these bootstrapping ideas as a natural extension and formulation of our earlier work, and as a means to continued progress in p.r. research. This leads to our concept of a complete system for p.r. research and to the methodology for the experiments we describe later.

For both generation and recognition of data, we devised a simple clustering language that we explain and give examples of in Section VI-B. We also describe how the human recognition tasks are "programmed" for the visual experiments.

A. Development of New Methodology

One of the most advanced forms of technology we have today is the man/machine partnership, if both the man and the machine are advanced. The man must be highly intelligent and sensitive, and the machine must be equipped with high performance interactive graphics, and it must represent the best that we know in machine computation. Also, the interface between man and machine must be natural and responsive in both directions. In a well-designed man/machine system, the machine complements man and converses with him. It explicitly mirrors for him his own subtle actions, so that he becomes more aware of the details and processes by which he performs his human recognition functions [118]. Only when man can understand his own subconscious or intuitive processes can he describe the unknown (subconscious recognition processes) in terms of the known [119].

Preceding page blank

Specifically, if he can describe these recognition processes in FORTRAN or in some other computer language, then he can relinquish the task (i.e., process) that formerly only a human could perform and can program his thought processes into the machine [120].

One of the main contributions of Zahn [52] has been his introduction into the formal p.r. literature of the word "gestalt," although he did not explain his motivation in depth, even in personal discussion with him. The importance of this concept in p.r. has been almost neglected. The originator of gestalt therapy, Dr. Fritz Perls, has deeply studied perception psychology and has considerably influenced the trend of modern therapeutic practice by means of conversational interaction between conflicting aspects of the patient's personality [121]. This type of conversational interaction is a valuable model when considering man/machine conversational interaction.

At SRI, we have been studying not only the psychology and engineering of the man/machine partnership [122], but also, we have been applying this to such practical problems as the development of better algorithms for machine p.r. The explicit statement of this research methodology is one of the most significant contributions we feel we can make to new methods in p.r.

One of the reasons we discuss the man/machine facility at such length here is that we observe that its importance to the development of, and benefits for, p.r. systems is not generally realized. We hope to show here that suitable interactive computing software is essential for advanced development in this unfortunately sophisticated, software field. We rely on suitable advanced interactive graphic facilities, as we explain here.

The general benefits of man/machine interaction have been pointed out by many writers in such diverse fields as architecture, management information systems, strategic planning, and many other applications [123]. In previous research, we have explored and benefited from man/machine interaction in developing p.r. and clustering algorithms [124].

There are several ways to view the man/machine interaction process. We can consider the machine algorithm or p.r. system as being trained by the man; i.e., the system "learns" from a human teacher. The teacher observes the performance of the p.r. system at each detailed step of the algorithm, using a CRT window into the algorithmic process. Note that in this overall process the system teaches or trains the man to understand in depth how the algorithm works.

It is essential to see and know the difference between what the man can recognize and what the algorithm can recognize. It is necessary to be on-line with a CRT window because without a window we cannot "see" into the details of the algorithm, and without being on-line, we cannot conveniently and feasibly try out tentative minihypotheses [125]. Each minihypothesis may involve only a few seconds of turnaround time. On the other hand, using batch mode computing, we can try out only a few major hypotheses because if the turnaround time from question to answer is long and complex (i.e., interrupted by other irrelevant activities, as is always the case in batch processing), the significance and subtle fitting of the hypothesis and the logical thread of the argument of the man/machine conversation may be lost [126].

The whole process of man/machine interaction is similar to human conversation on topics that are difficult and complex to understand. The evolutionary process in man/machine research can allow the development of complex procedures or algorithms for recognition that would otherwise take a much greater time or perhaps would never evolve [127].

1. The Concept of a Complete Pattern Generation and Recognition System

In a typical p.r. application problem, the data comes from some remote application measurement system [128]. Thus, we usually have little or no control over the data. To fully understand the workings of a p.r. system, it is necessary to understand the data generation process, the configuration of the measuring equipment, and the viewpoint of the data collection equipment at the time that the patterns are collected. A very effective way to achieve this type of system overview is to generate artificial data and to recognize these patterns in a kind of closed-loop system, in which we have full control of all the data-generating parameters. In this way, we can work from a mathematical model of the p.r. system, write computer algorithms that implement this model, and control the data sampling process. We can then feed through to the p.r. algorithm as much or as little of this model information as we wish. The more a priori knowledge of the model we supply to the recognition algorithm, the easier it is for it to achieve error-free performance. Conversely, if we only supply a small amount of information to the recognition algorithm, it will have to "learn" a lot, make many guesses, try tentative hypotheses, and experience high error rates--at least in the initial stages of the "learning" process. Therefore, in our experiments on p.r. using the PDP 11 computer and display, we generate data from a model and use this data to exercise the p.r. algorithm.

In a typical p.r. application, the choice of data observation instruments or transducers will affect the nature of the recognition. For example, if the source of patterns is generating wide-band signals, and if the measuring instrument is a narrow-band one, the source of signals will appear to be narrow-band because that is the only manner in which the transducer can respond. For accurate observations, therefore, a "wide-open" measuring system is required. In psychological terms, the

observer must be free of preconceived notions and biases to perceive the reality of his environment. Any form of observer bias will distort his perception of the environment so that he will not be able to see the environment as it is [129]. This same argument applies to any recognition algorithm when it gives undue weight to an irrelevant factor in the data. To measure and test recognition performance, and to discover the relative importance of various factors in the data, we developed the approach of a closed-loop recognition and generation system and the following experimental methodology.

2. Experimental Methodology

In our experiments, we test the hypothesis that the current computer p.r. algorithm (Version A) works well on the current test data set (Set A). The meaning of "works well" is that the machine algorithm performs at a certain level of error compared to a human pattern recognizer. Thus, to measure the performance of the computer algorithm, we must ultimately compare the results from a human. To compare man and machine, we must have them perform similar tasks and give outputs in a comparable format. The input data to man and machine must also be similar. Our experiments start at a simple level, using easily described data to begin with, but we carefully investigate the accuracy of the responses and the detailed nature of the judgment criteria.

If any discrepancies arise between man and machine recognition performance, then we may modify the algorithm or the experimental procedure used for the human task. Once satisfactory closure or cross-comparison of results has been obtained with data Set A, we may proceed to data Set B, which has more complex characteristics, to see whether the algorithm can keep pace with human performance. If it cannot, we may try to improve the algorithm, using our insight into the human recognition task as a guide to the required modification.

To make the human performance measures independent of personal idiosyncrasies, we have to include many human observers in the perceptual experiments.

We tested the human and machine recognition only on samples of data drawn from an environment or model of a general data-generating situation. Initially, we controlled the data generation closely, so that we could absolutely verify the accuracy of performance of the recognition systems. Later, we may use data from an uncontrolled source, such as cloud measurements from satellites, for testing the algorithms in an area of practical application.

The purpose of the first experiment was to test the concept of a cluster center. The data-generating model for this experiment was a single point on a plane. Samples were generated from this point by adding noise to the horizontal and vertical components of the data coordinates. This simulated the measurement process in which errors of observation are always present in the classical sense of the theory of observations [130]. The task of the p.r. systems (both human and machine) was to describe the data generated and presented from this single representative point, i.e., cluster center. The situation is represented in Figure 38.

The FORTRAN computer programs for performing this on the PDP 11 are described briefly as follows. A random number generator provides (floating point) random numbers, uniform in the range from zero to one. From these a Gaussian random number can be generated.

The program can be controlled to generate a desired number of points and to display these, both on the CRT and, less accurately, on the printer. The program will accept the cluster center at the location indicated by an observer using the display cursor. The machine p.r. algorithm, in this case, is the one-cluster mean value.

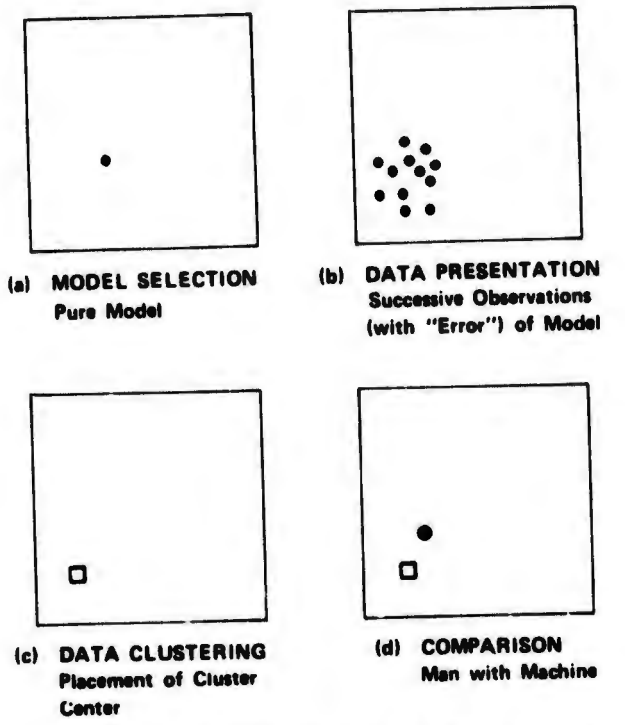


Figure 38. Simple Sketch of Experimental Steps for Testing the Concept of the Cluster Center

A suitable null hypothesis in this experiment is that there is no essential difference--over a large number of trials and for wide variations in the nature of the error--between the mean value found by the chosen p.r. algorithm and the human observer's selection of a center. If there is no essential difference, then the method of computation of the cluster center by the algorithm is validated. If there is a difference, we have to discover what the difference is and then modify the p.r. algorithm to perform according to the human visual gestalt formation [131].

B. The Clustering Language for Generation and Description of Data

In applying the above principles in practice, we implemented a particular computer language [132] to allow convenient generation and recognition of data. By recognition, in this context, we mean the recovery of a data description that corresponds to a data generation language. Thus, recognition performance can be evaluated by comparing the generating language with the description language.

The development of this clustering language is necessary for two main reasons:

- To allow the comparison between recognition and generation, both by human and by machine, and to allow evolutionary development of algorithms in a closed-loop bootstrap process.
- To provide a practical and convenient way to generate data and to provide a means of interface between the CDC 6400 and the PDP 11, since no other hardware or software interface exists. (The interface is necessary because each machine has different features that we wish to use.)

The commands shown in Table III are implemented in the system and can be input either from the on-line Teletype on the PDP 11 or from the card reader on the PDP 11. In addition, the same cards may be fed into the CDC 6400, which operates only in a batch mode. Each command starts with a single character and may be followed by numerical parameters. We can express such syntax generally by means of the syntax equations:

COMMAND = alpha/alpha parameters

ALPHA = any alphabetic character

PARAMETERS = numerical values.

This is a very simple language, but it is sufficient to illustrate the methodology and to allow great conveniences in practical use. We later discuss extensions of this approach that we feel are valuable topics for further research.

TABLE III. COMPUTER COMMANDS FOR THE CLUSTERING LANGUAGE

Number	Commands (operator options)		Default values
1	A		
2	B		
3	C	Cluster center for Cluster I	(0,0)
4	D	Standard deviation for Cluster I	(300,300)
5	E	Exit to monitor	
6	F		
7	G	Go-start execution--generate new cluster centers	
8	H	Use old cluster centers (assumes no changes to specs)	
9	I	Cluster center index	1
10	J		
11	K		
12	L	List value of parameters of Cluster I on Teletype	
13	M		
14	N	Number of cluster centers	1
15	O	List the commands on line printer (LP)	
16	P	Number of points in Cluster I	100
17	Q	Debug print if Q $\neq$ 0	0
18	R	Initial random number value--two-integer values	
19	S	Switch from card reader to Teletype; or vice versa	Teletype input
20	T		
21	U		
22	V		
23	W	Rewind data file	Start of file
24	X		
25	Y		
26	Z	Zero summary statistics	

1. Excerpts from Typical Printouts

a. Example 1

Table IV shows a Teletype print of the status of the cluster specifications obtained by typing "L." In this example, Cluster 3 is listed first because that cluster's specifications had just been amended.

TABLE IV. TELETYPE PRINTOUT SUMMARIZING CLUSTERS DISPLAYED

```
L
CLUSTER  3    CC      Ø.-1ØØØ.     DEV  5Ø.   5Ø.
I1 L
CLUSTER  1    CC -1ØØØ.     Ø.     DEV 2ØØ. 1ØØ.
I2 L
CLUSTER  2    CC  1ØØØ.     Ø.     DEV 1ØØ. 2ØØ.
```

b. Example 2

Figure 39 shows a line printer plot for four clusters, each having the same number of points and deviations. Due to the low resolution of this plot, some points on the printout may represent several data points because they fall into the same print position. An actual photograph from the display screen would illustrate the full detail possible with the CRT display. The PDP 11 printout also shows the theoretical specifications in the user language and the viewer's response in placing cluster centers, as shown in Table V.

2. Procedure for Conditioning Human Subjects in Visual Judgment Experiments

It is necessary to instruct or otherwise condition the subjects taking part in the visual judgment experiments on the PDP 11 and to be consistent in giving these instructions so that subjects are not biased

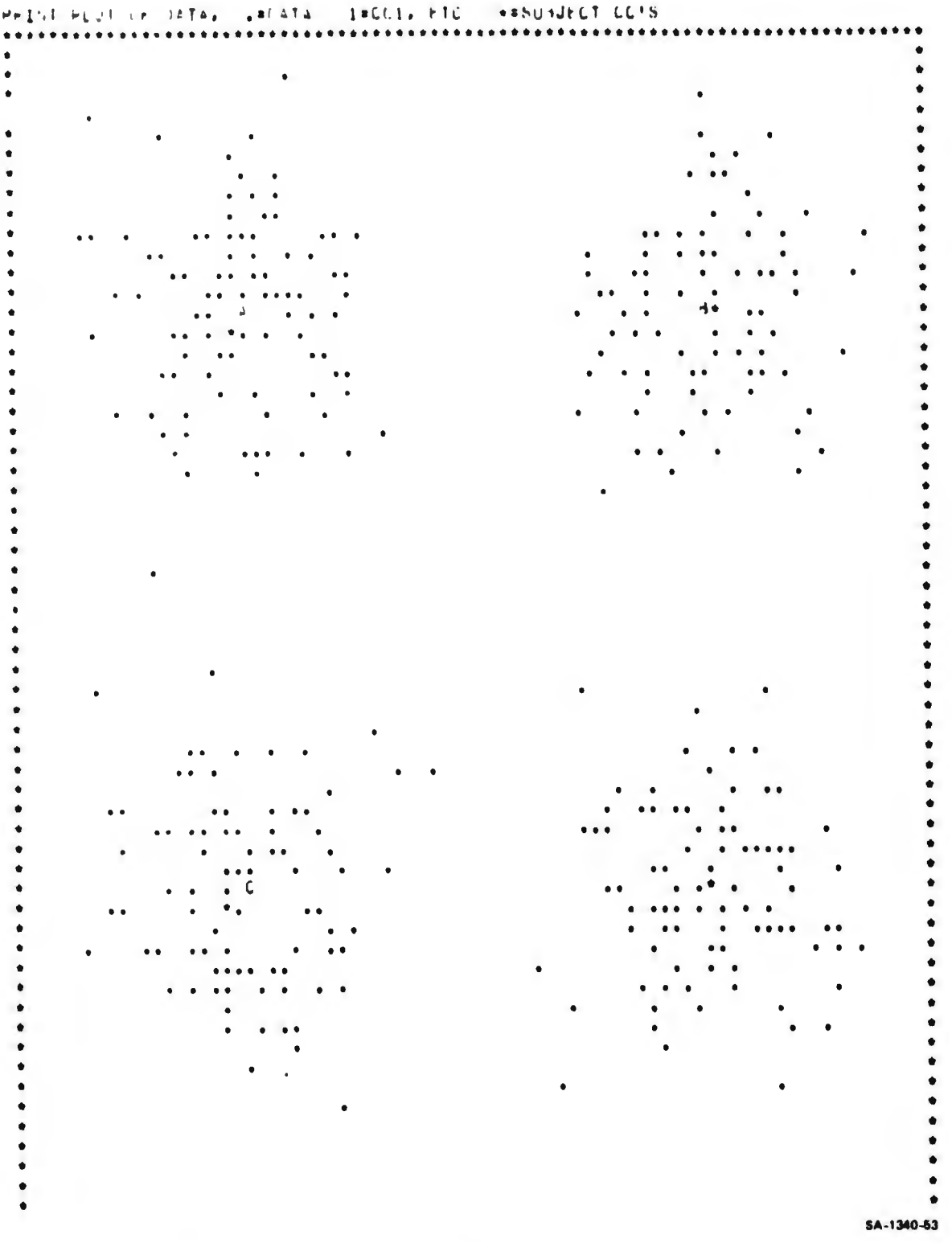


Figure 39. Print Plot of Data from PDP 11 Line Printer

TABLE V. TELETYPE PRINTOUT OF THEORETICAL, GENERATED, AND PERCEIVED CLUSTERS

THEORETICAL

NUMBER CLUSTERS 4 BEGINNING RANDOM NUMBER 20864 -17109

OPERATOR SPECS					AS GENERATED				
		CENTER		STAN DEV			CENTER		STAN DEV	
CLUST	NPTS	X	Y	X	Y	NPTS	X	Y	X	Y
1	100	-1000.	1000.	300.	300.	100	-1010.	1008.	306.	317.
2	100	1000.	1000.	300.	300.	100	1049.	1007.	295.	288.
3	100	-1000.	-1000.	300.	300.	100	-1017.	-987.	303.	309.
4	100	1000.	-1000.	300.	300.	100	1055.	-985.	314.	291.

SUBJECT RESPONSE

NUMBER CLUSTERS 4

	CENTER		CLOSEST TO			DIFFERENCE		
CLUST	X	Y	NO	X	Y	X	Y	DIST
1	-1035.	910.	1	-1010.	1008.	25.	98.	102.
2	1075.	960.	2	1049.	1007.	-26.	47.	53.
3	1030.	-1000.	4	1055.	-985.	25.	15.	29.
4	-1095.	-1085.	3	-1017.	-987.	78.	98.	126.

differently by means of the task descriptions [133]. Therefore, we prepared our instructions carefully and provided a clear and consistent input to each of the subjects. Since reading is analogous to compiling, and since instructing for the performance of a task is itself an algorithmic task, we call our formal instructions to the subjects, a computer program for human experimental subjects. It is well known in the field of experimental psychology [134] that these precautions and this consistency are necessary [135]. Although instructions to the subjects may not be crucial in these particular experiments, they would definitely be crucial in some experiments, and we wish to establish our methodology beyond reproach so that it can be uniformly extended to more crucial experiments.

Table VI gives an example of one of these programs. It would be interesting to show how two different programs such as this can give rise to a significant bimodal response in experimental results, using a large number of subjects viewing identical data. However, such a demonstration is beyond the scope of the present project, except to note that the phenomena of visual illusions also have this bimodal quality [136, 137].

3. An Example of the Use of the Clustering Language

To illustrate the use of the clustering language, the following simple example uses the methodology of closed-loop generation and recognition of data. For illustrative purposes, the data in this example are simple to generate and recognize.

The first step is the generation of the data by means of the input language specifications. We wish to generate a large cluster on the left-hand side of the screen and a small cluster on the right-hand side. Both clusters will be on the same horizontal axis through the middle of the screen.

TABLE VI.

COMPUTER PROGRAM FOR HUMAN EXPERIMENTAL SUBJECTS

24 April 1972 Version 1 Programmer: D. J. Hall

Introduction

This is a high-level language program, which you, as an experimental subject "machine," must compile to participate in a pattern-recognition experiment. Please read it carefully, and indicate whether you have compiled (i.e., understood) it. If you have questions, please write your output (syntax error messages) on a sheet of paper. Verbal discussions will be avoided because we wish to record all transactions between the experimental subject and the high-level software monitor system (i.e., the experimental methodology). Your impressions, reactions, and remarks will be welcomed as valid output comments. However, the criterion for evaluating your compilation performance will be task accomplishment; so the purpose of this program is to prepare (i.e., program) you for the following task.

Task Description

You will be asked to sit in front of a CRT display screen, and you will be shown how to use the display cursor ("mouse") to indicate points on the screen.

Various sets of data will be displayed on the screen in front of you You are to view the data and place cluster centers in the data with the mouse. Once you press the mouse button, it is preferable not to change your selection. However, if you misplace a cluster, you can correct it, using one of the other mouse buttons. Each data set will be presented several times. Your first response should place the best single cluster in the data, your second response should place two clusters in the data, and so forth. At the end of the sixth presentation, you may be asked to name the number of clusters that give the best fit.

Discussions between the experimenter and subject should be avoided, but an essential communication can be recorded as a formal message on paper.

Your Output

Have you compiled these instructions (Yes/No)? If not, please write your queries down as clearly and simply as you can. These syntax error messages will be used to debug this program or your compilation of it.

This natural language description of the data we wish to generate is not specific enough in either detail or format to input to the computer system to generate suitable data. However, it is interesting to speculate that if we make some reasonable assumptions about the meaning of "large cluster," "small cluster," and other terms, such as "left-hand side of the screen," then it might be possible in future research work to construct a compiler for producing appropriate data from such a fuzzy input language specification.

For our example, we can easily choose some specific numbers to represent the sizes of clusters and positions on the screen. The statements in Table VII cause the generation of two clusters, shown in Figure 40. (Refer to the list of commands in Table III for the detailed interpretation of each item.)

TABLE VII. TELETYPE PRINTOUT SPECIFICATIONS FOR CLUSTERS SHOWN IN FIGURE 40

```
 N2 I1 C -1000 0 D 300 300 P 300 L
CLUSTER 1 OF   2     NPT 300      CC -1000.    0.    DEV 300. 300.
 I2 C 1000 0 D 60 60 P 60 L G S
CLUSTER  2 OF  2     NPT  60      CC  1000.    0.    DEV  60.  60.
```

Note: The second and fourth lines are caused by the command "L" to list the statement after compiling it.

The data shown in Figure 40 may be generated either on the CDC 6400 or on the PDP 11; identical numbers will be produced by the random number generators on each of these machines. The actual generated values will differ from the theoretical values specified in the input language because of the small-sample restrictions. The program prints these values as shown in Table VIII. (The number in the external right column is the correlation coefficient between X and Y.)

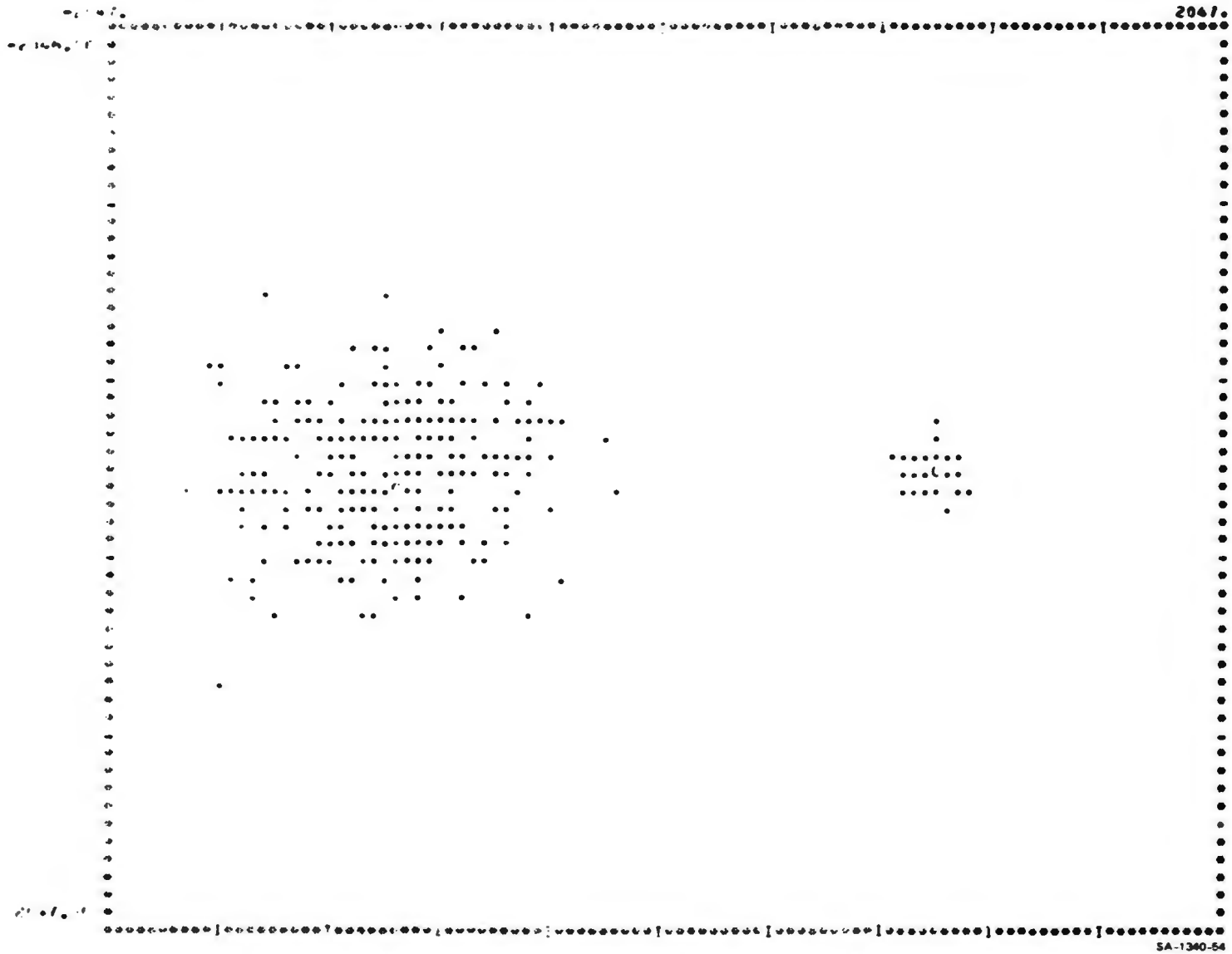


Figure 40. Line Printer Plot of Large and Small Clusters

TABLE VIII. TELETYPE PRINTOUT OF THEORETICAL AND GENERATED CLUSTERS

THEORETICAL

NUMBER CLUSTERS 2 BEGINNING RANDOM NUMBER 5103 2187

	OPERATOR SPECS					AS GENERATED				
	CENTER		STAN DEV			CENTER		STAN DEV		
CLUST	NPTS	X	Y	X	Y	NPTS	X	Y	X	Y	
1	300	-1000.	0.	300.	300.	300	-1006.	6.	296.	298.	-.1006
2	60	1000.	0.	60.	60.	60	1015.	-8.	59.	66.	-.0350

The display shown in Figure 40 is a line printer plot with the usual spacings of ten characters per inch along a line and six lines per inch measured vertically. Therefore, the resolution of this display is severely restricted, since all points are normalized to the position of the period character, and multiple points in the same character space are represented by only one period. These printer plots conveniently illustrate the general shape of the clusters but do not accurately portray the density of points if it is greater than one point per print position. For general visual interpretation, this crude display is usually preferable to a corresponding table of numbers. From the PDP 11, the actual values can be displayed on the CRT with high precision.

Clustering of the data generated may be done either on the CDC 6400, by means of the ISODATA algorithm, or else on the PDP 11, using human visual judgments to place cluster centers in the data by means of the display cursor device, or mouse. The ISODATA output gives various statistics in addition to the cluster center positions and their standard deviations. This form of output has been described in previous reports [138, 125]. The additional output now generated by the algorithm is in the same form as the input data-generating language, and for this example we obtain the output language from the printout shown in Table IX.

TABLE IX. ALGORITHMIC OUTPUT FROM CDC 6400

```
N  2
I  1  P300   C-1006.    6.   D   296.   298.
I  2  P 60   C 1015.   -8.   D    58.    66.
```

A comparison may now be made between the input language and output language, and it can be seen that the ISODATA algorithm with descriptive output gives an accurate clustering language description of the input. The output can also be compared to the actual data generated rather than to the ideal (theoretical) specification of the parent population clusters. From this second comparison, we can see that the two data descriptions are identical (except for hardware differences between the two machines).

The above example illustrates the closed-loop generation and recognition process for a simple example. In dealing with more complex data sets, the output language description may not match the input specification so well. To illustrate the discrepancy of the recognition in these cases, it is helpful to generate and display a second data set from the clustering output. This second display can then be compared with the first display of the generated data to see in what ways the two data sets differ.

SECTION VII

APPLICATION OF THE GENERALIZED DISTANCE OF MAHALANOBIS

This section defines and discusses the generalized distance of Mahalanobis, together with a new lumping algorithm that must take into account the fact that the distance from A to B is not equal to the distance from B to A in an anisotropic (perceptual) space. We then give experimental results (Section VII-B), which apply this new lumping algorithm using generalized distance, the clustering language, and the experimental methodology to both human and machine clustering problems.

The relevance of our research experiments using clouds or clusters of data generated artificially is to meteorological applications in which we study satellite data from clouds. In Section VII-E, we discuss other properties of satellite cloud data that make it a suitable application problem for testing clustering algorithms. In Section VII-F, we consider cluster characteristics other than position that contribute to the distance criterion for lumping clusters or points together.

The idea of a generalized distance in a statistical field was introduced by Mahalanobis for a multivariate normal population. In general, for an n-dimensional normal population, the density function is given by

$$p(\underline{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{\frac{1}{2}}} \exp\left[-\tfrac{1}{2}(\underline{x} - \underline{\mu})^{t}\Sigma^{-1}(\underline{x} - \underline{\mu})\right]$$

where $\underline{x}$ is an n-dimensional column vector, $\underline{\mu}$ is the n-dimensional mean vector, Σ is the n-by-n dimensional covariance matrix, and $|\Sigma|$ is the determinant of Σ. Samples drawn from a tightly structured population of this description cluster around the mean values $\underline{\mu}$. The shape of the distribution around the mean is given by the covariance matrix. The Mahalanobis distance, m between $\underline{x}$ and $\underline{\mu}$ in this field is given by

$$m^2 = (\underline{x} - \underline{\mu})^{t}\Sigma^{-1}(\underline{x} - \underline{\mu})$$

The contours of constant density are hyperellipsoids, such that any dimensionless point on a contour has the same Mahalanobis distance to the local mean. In general, this statistical field can be considered anisotropic, and it represents a warped space that has nonuniform properties that vary, depending upon the location being considered in the field. The shape of the field or warping of the space depends on the data in the field.

Although we cannot program these matrix formulations directly in a computer language such as FORTRAN, we can translate the mathematical formulations into an algorithm to compute such quantities as the covariance matrix Σ. However, for multivariate data, this requires storage of a large array of floating-point values and significant associated computation time. Other objections to the estimation of Σ were discussed in Section II-C. In ISODATA, the covariance matrix is not available, and we choose to approximate this distance computation as shown below, considering the two-dimensional case. (The computer program will handle up to 50 dimensions.) The distance of a dimensionless point (x, y) to a cluster center at $(\bar{x}, \bar{y})$ with coordinate sample deviations (S_x, S_y) is given by

$$m^2 = \frac{(x - \bar{x})^2}{S_x} + \frac{(y - \bar{y})^2}{S_y}$$

The sample deviations, S, are already available in the ISODATA program.

For a point that is not dimensionless, we must take its own deviation into account. This deviation is related to the precision of the measurement that establishes the position of the point. If the measurement is highly precise, the deviation is very small. If the point is remote from other points, it is not likely to be part of another cluster since its isolation will be reliable and significant because of the low error in the measurements. We next consider how to use the anisotropic generalized distance in the lumping algorithm of ISODATA.

A. Lumping Algorithms

If ISODATA is used with the generalized distance, m, as the measure of relationship, the distance between two clusters is not the same in each direction. This confuses the question of which pairs of clusters to lump.

A more fundamental consideration is whether the clusters should be lumped. However, this question can be viewed as the inverse of the question: Should one cluster be split into two? This question can be handled by the Kolmogorov-Smirnov test, as applied in Section IV-B. We assume that the same test can be applied to decide the question of whether or not to lump or to split clusters.

Assuming that this fundamental question is settled, we then ask, which cluster pair should be lumped? Before answering this question, we first review the algorithm we have used for several years.

1. The Euclidean Distance Lumping Algorithm

This algorithm uses the Euclidean distance table (EDT) and finds the closest pair of clusters to lump. This is evidently simple but we now need a refinement based on the generalized distance table (GDT). The EDT has symmetry about the diagonal because if d_{ij} is the distance from Cluster i to Cluster j, then $d_{ij} = d_{ji}$, and finding the closest distance is a simple ranking procedure.

2. Generalized Distance Lumping Algorithm

First, consider the simple case in which there is clearly one pair of clusters much closer to each other (using generalized distance) than all the rest, as illustrated in Figure 41. For this figure, bear in mind that compact, dense clusters are associated with a larger

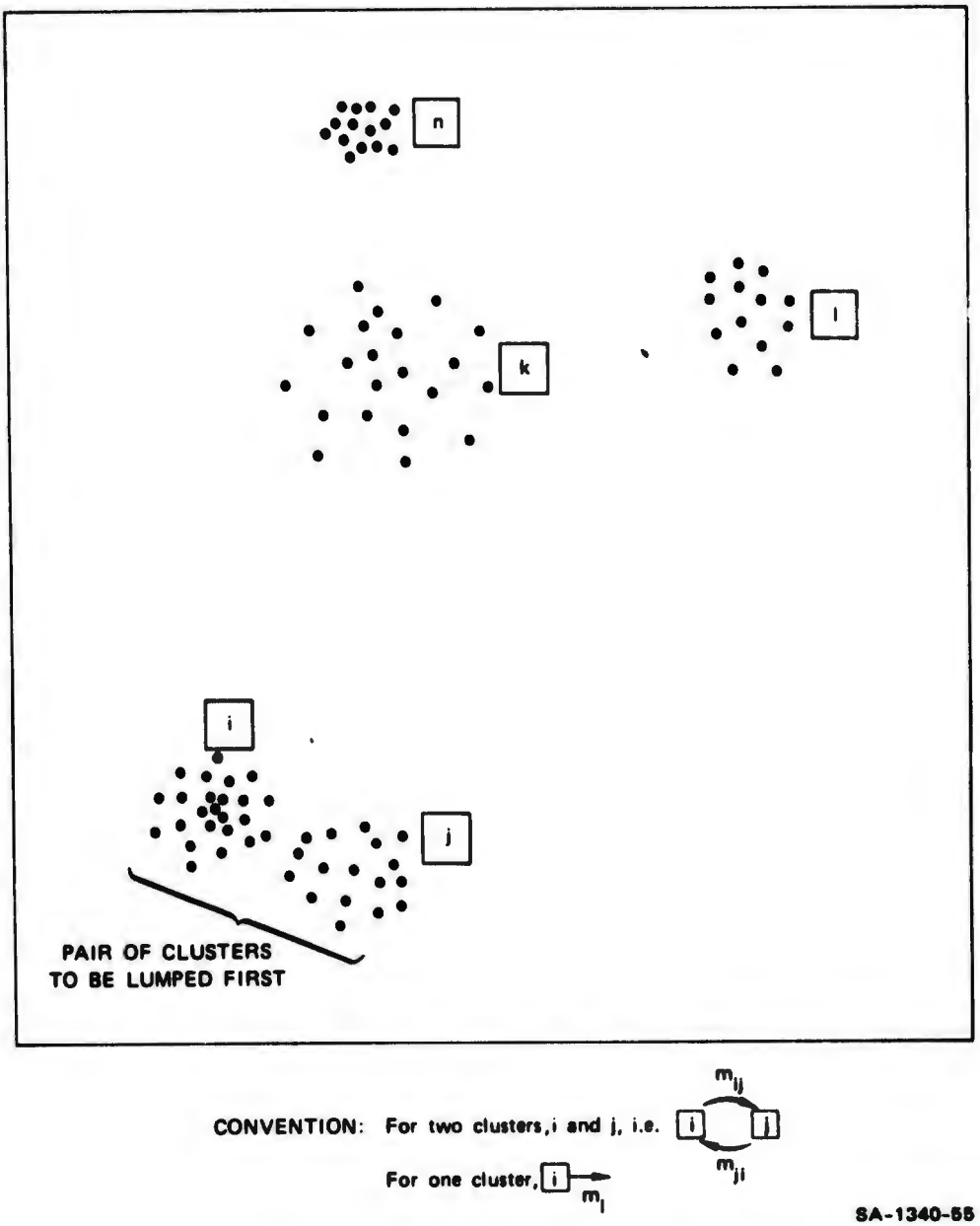


Figure 41. Case in Which the Pair to be Lumped is Evident

generalized distance, m, than are large, sparse clusters. Thus, $m_{ij} \neq m_{ji}$. Consider the GDT for this case, as shown in Table X.

TABLE X. GENERALIZED DISTANCE TABLE FOR FIGURE 41 DATA

	i	j	k	ℓ	n
i		m_{ji}	m_{ki}	$m_{\ell i}$	m_{ni}
j	m_{ij}		m_{kj}	$m_{\ell j}$	m_{nj}
k	m_{ik}	m_{jk}		$m_{\ell k}$	m_{nk}
ℓ	$m_{i\ell}$	$m_{j\ell}$	$m_{k\ell}$		$m_{n\ell}$
n	m_{in}	m_{jn}	m_{kn}	$m_{\ell n}$	

If we rank all the values in this table from the minimum to the maximum, then, because the need to lump Clusters i and j is evident from Figure 41, the ranking of distances (m), by construction, will begin with m_{ij}, m_{ji},..., or m_{ji}, m_{ij},... .

Thus, we can see that the simplest form, or outer loop, of the generalized distance lumping algorithm must be:

- Rank the GDT values.
- If the first two (lowest) distances involve the same two clusters, then
- Lump these clusters, else
- ?

Note that this is consistent with the Euclidean lumping algorithm as a special case. We now have to determine what to do if the case is not so simple. There are several alternatives.

The first is to find the first matching pair in the rank list. For example, if the rank list is m_{jk}, m_{kl}, m_{no}, m_{lk}, m_{jp},... the lowest matching pair in the rank is m_{kl} and m_{lk}.

The second alternative is to use an average or pooled value or, in general, some function derived from the two measures, essentially converting the bilateral distance (m) into a unilateral one (m'); i.e.,

$$m'_{ij} = fn(m_{ij}, m_{ji}) = m'_{ji}$$

Then we would use these values in an EDT, as with the Euclidean lumping algorithm. The simple average value, as a possible example function, is given by

$$m = \frac{m_{ij} + m_{ji}}{2}$$

The third alternative is to use the minimum of the maximum of the distances in the GDT to implement the same pair selection procedure as in the first alternative. The third alternative requires less storage for the computation than the first alternative because the evaluating function, i.e.,

$$\min_{\text{all } i,j} [\max(m_{ij}, m_{ji})]$$

can be computed "on the fly," whereas the procedure in the first alternative requires storage of the values prior to a ranking. This procedure essentially accomplishes the ranking at the time that the distances m_{ij} are computed.

Note that the GDT represents about twice the computation required for the EDT because the distances are bilateral instead of unilateral. This GDT algorithm has been programmed in FORTRAN, and it provided the experimental results shown below.

B. Experimental Results Using Generalized Distance

We now consider two data types and some experimental clustering results typical of many results with these data types. In Section V-A-2, we gave examples of inadequate clustering results using Euclidean distance. We discussed the cases illustrated there in Figures 30(a) and 30(b).

1. Clustering the Unbalanced Dumbbell Data

We first consider the unbalanced dumbbell data, a large cluster and a small one close to it. The large cluster, shown in Figure 42, has 100 samples, and the smaller one has 20. The circular deviations specified in the clustering language are 300 and 60, respectively. The algorithm partitions the clusters well, except for one outlying point of the large cluster. This point is quite close to the small cluster and might be considered as a wild shot. Whether or not it should belong to the larger or smaller cluster is, in fact, a matter for human judgment, but the error of the algorithm is not significant. We have sketched an approximate boundary between the two clusters. It is interesting to note that the distance from the larger cluster to the smaller is computed as 3.4 (relative units), and from the smaller to the larger, it is 15.2.

If we consider the linguistic specifications given automatically by the program output, we can make a more quantitative comparison. Cluster I1 is the larger one. Table XI gives the input specifications or display language for generating the data first, then the actual data generated, and finally, the clustering result.

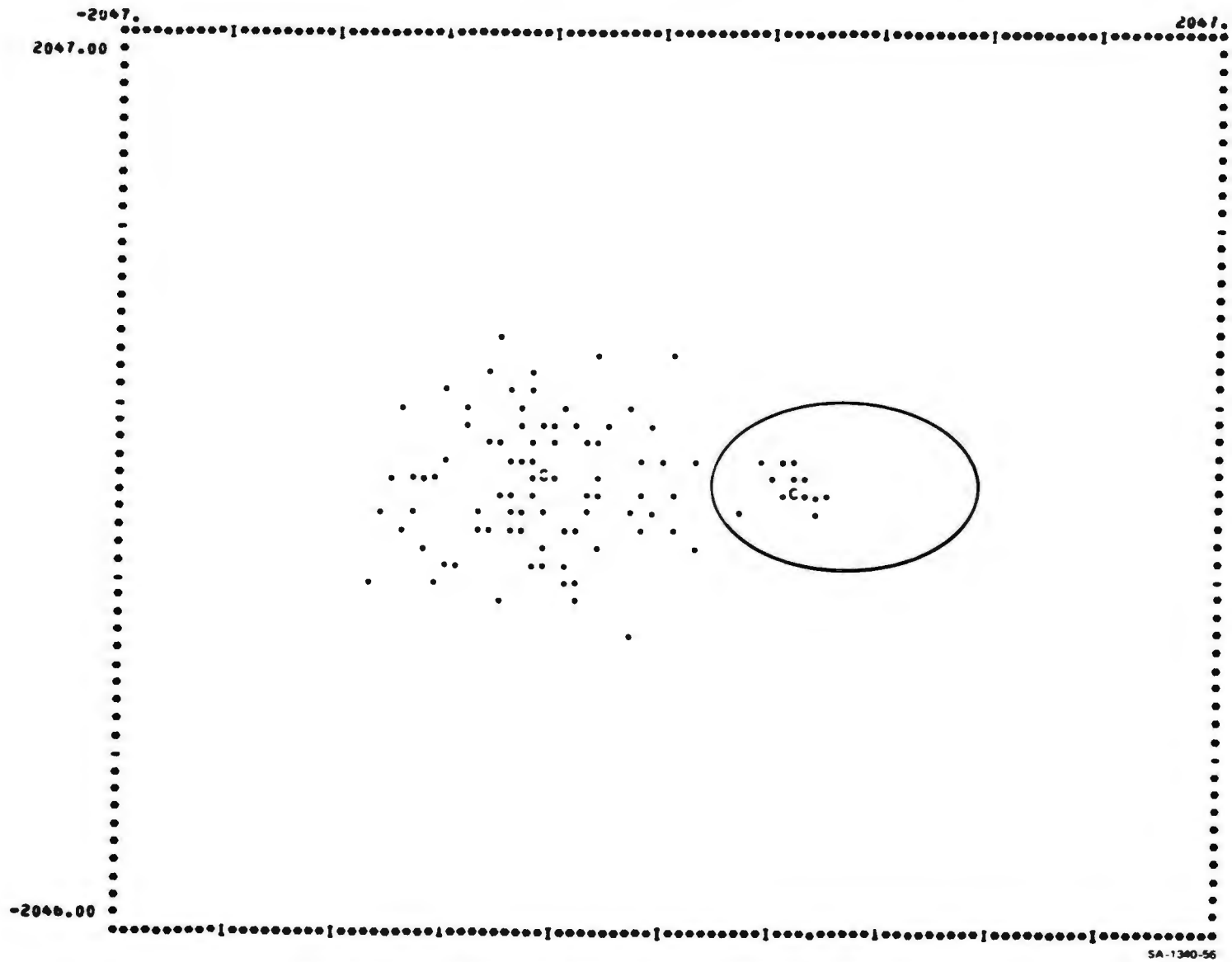


Figure 42. Printer Plot and Clustering of Unbalanced Dumbbell Data

TABLE XI. SUMMARY OF EXPERIMENT ILLUSTRATED IN FIGURE 42

INPUT	N2	I1	P	100	C	-500	0	D	300	300	I2	P	20	C	500	0	D	60	60
GENERATED	N2	I1	P	100	C	-476	14	D	297	287	I2	P	20	C	493	-11	D	58	56
CLUSTERED	N2	I1	P	99	C	-485	13	D	285	287	I2	P	21	C	485	-6	D	64	57

The differences among these three data descriptions are within several percent only; so the performance of the algorithm is not in conflict with human gestalt clustering and description of this data. In this case, the clustering produced a remarkably good description of the input data.

We now consider another example of the same type of data that has a larger number of points, namely, 300 in the larger cluster and 60 in the smaller. The separation between the centers is, however, the same 1000 screen units. The printout of this data is given in Figure 43. As explained earlier in Section I, we cannot display the full resolution of the data in the printer plots we give. Table XII gives the Euclidean distance results to show that they are in error, and the generalized distance results, which are in perfect agreement with the generated data. In this example, no wild shot appeared.

From these linguistic statements and the positions of the cluster centers in Figure 43, we see that the ISODATA algorithm has been significantly improved by the use of generalized distance.

This method can even provide reasonable results for overlapping data, as in the case when the smaller cluster moves partly inside the larger one. Consider the example shown in Figure 44, in which the specified separation between cluster centers of only 400 units is almost as small as the 300-unit deviation of the larger cluster. The input, generated, and output linguistic specifications are given in Table XIII.

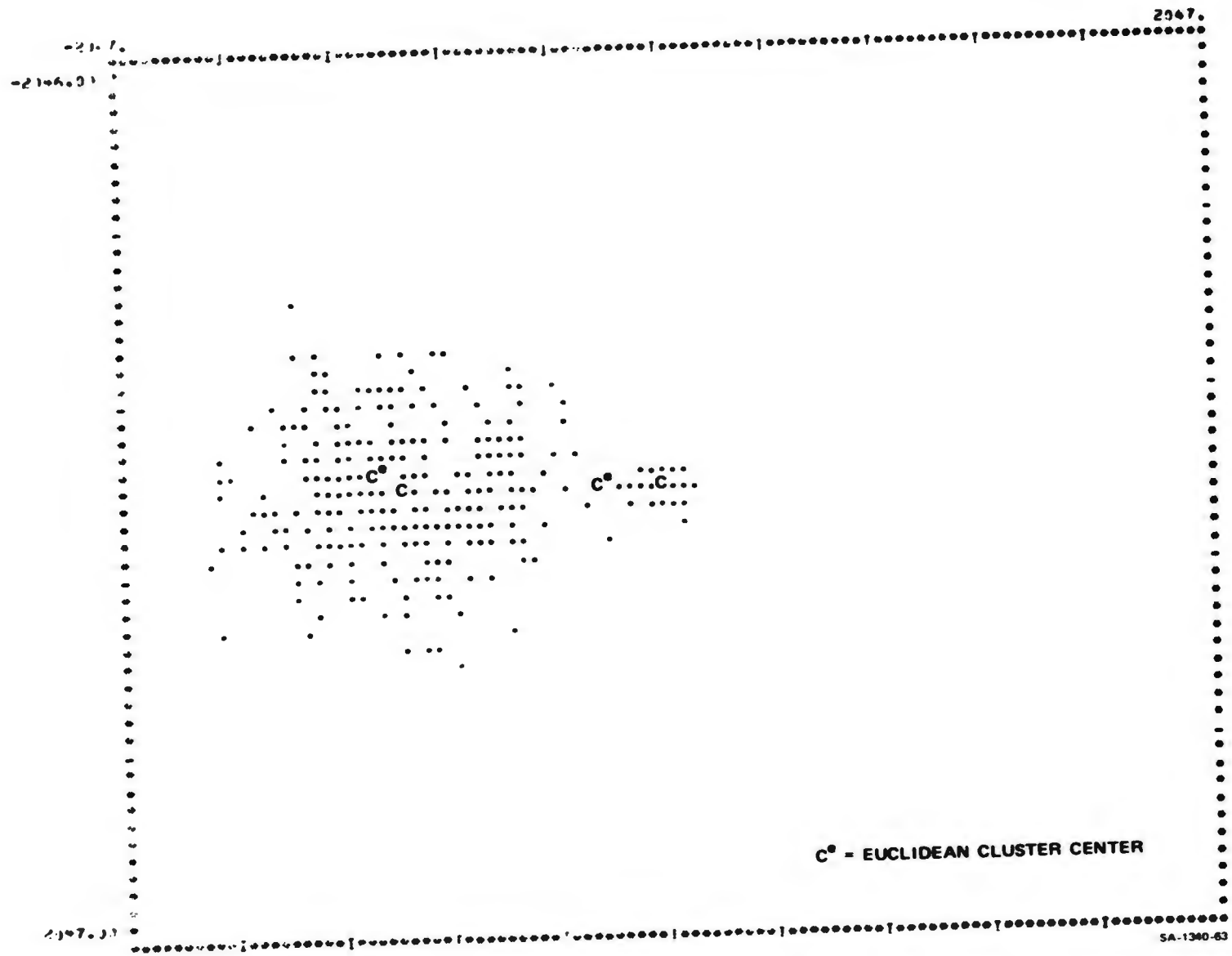


Figure 43. Clustering of Dumbbell Data by Euclidean and Generalized Methods

TABLE XII. SUMMARY OF EXPERIMENT ILLUSTRATED IN FIGURE 43

INPUT	N2	I1	P300	C	-1000	0	D	300	300	I2	P	60	C	0	0	D	62	60
GENERATED	N2	I1	P300	C	-993	-3	D	310	311	I2	P	60	C	-3	-1	D	62	55
EUCLIDEAN	N2	I1	P258	C	-1070	4.2	D	258	315	I2	P	102	C	-213	-19.9	D	262	181
GENERALIZED	N2	I1	P300	C	-993	-3	D	310	311	I2	P	60	C	-3	-1	D	62	55

We have previously used a statistic known as total squared error as an indicator of both the relative "clusteredness" of data and the convergence of the settling process. (We assume that the reader has some background in these topics [125, 105].) We notice that this statistic often increases after each settling partition, rather than decreasing as it does with Euclidean distance. As the clusters become more compact, the relative distance increases because the deviation gets smaller. Thus, the error, composed of the sums of squares of distances, gets larger. However, during some iteration of the algorithm, the increase in error is offset by the fact that the center is the closest possible point to all the sample points; the center becomes more central, which decreases the distances from the new center to each of the sample points. However, all the factors involved in this settling process are not sufficiently well understood and will require further research.

2. Clustering of Cigar Data

In this experiment, we clustered the so-called cigar data shown previously in Figure 30(b). No problem is encountered using Euclidean distance in clustering the two clusters represented by this data at the two-cluster level, if the clusters are far apart. We have verified experimentally that at a separation of 1000 units between the cluster centers, perfect clustering results using Euclidean distance. However, when the clusters are closer, the maximum variance is in the vertical direction,

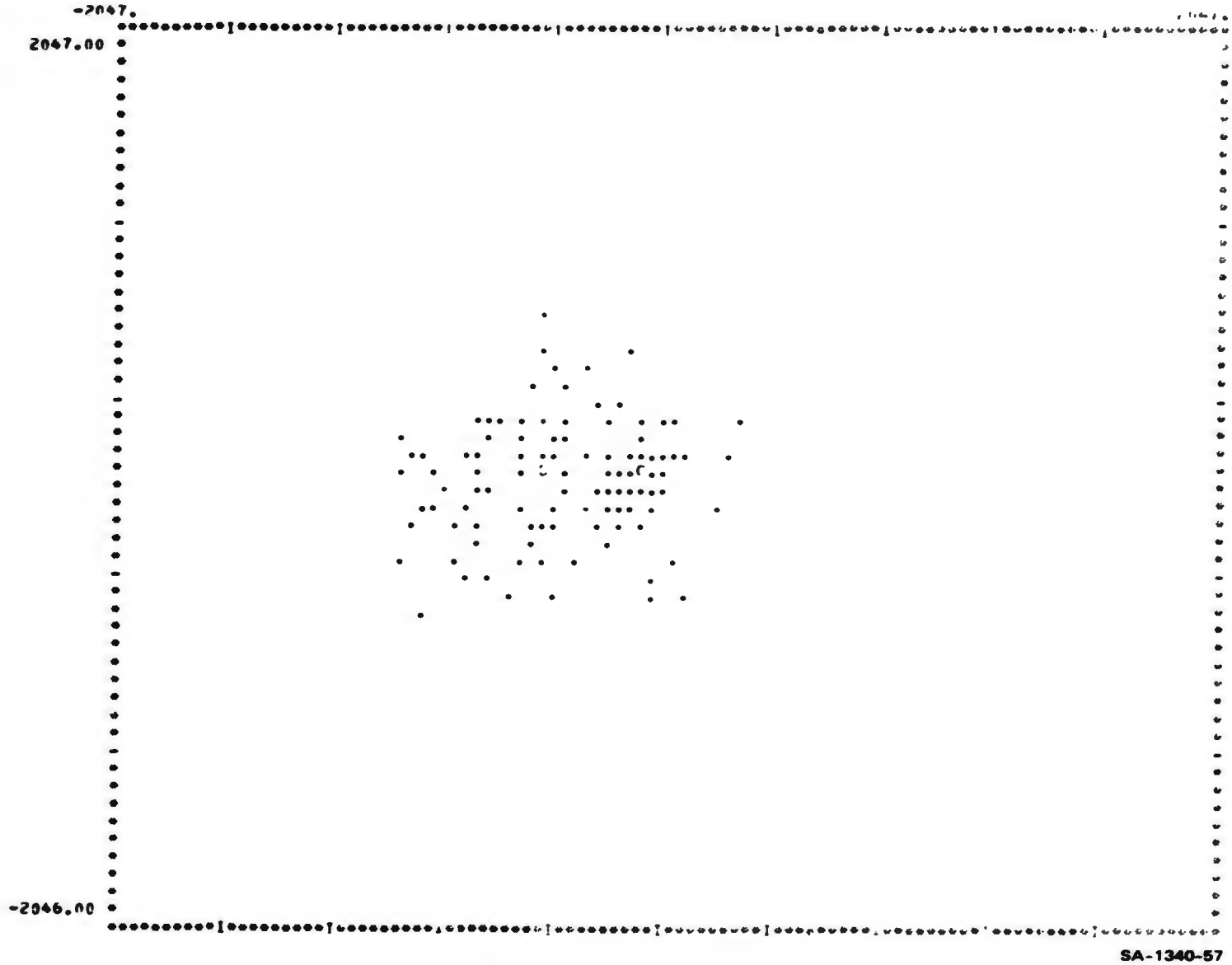


Figure 44. Clustering of Overlapping Data

TABLE XIII. SUMMARY OF EXPERIMENT ILLUSTRATED IN FIGURE 44

INPUT	N2 I1 P 100 C -500 0 D 300 300 I2 P 40 C -100 0 D 60 60
GENERATED	N2 I1 P 100 C -451 7 D 295 288 I2 P 40 C -105 0 D 68 61
CLUSTERED (OUTPUT)	N2 I1 P 97 C -465 3 D 287 289 I2 P 43 C -95 8 D 58 70

and splitting occurs as shown in Figure 30(b) because the algorithm splits perpendicular to the axis of maximum variance, which is vertical when the clusters are close. The program shows that this is a stable partition, with the cluster centers situated on the boundary between the visual clusters.

An example of such an unsatisfactory clustering is given in Figure 45. There are 300 points in each cluster, and the separation between specified centers is only 500 units in the horizontal direction. Visually, there are clearly two clusters, but the program using Euclidean distance does not find them, although it converges to a stable value.

If we introduce generalized distance, the performance of the program is improved, as illustrated in Figure 46. Note that these two data sets have different samples, but their clustering language or statistical specifications are identical. The visual cluster centers are found by the program, and this clustering is also stable. However, during iteration of the algorithm, in splitting up into a greater number (six) of clusters, some of the cluster centers locate either (1) some relatively isolated points, each of which remains as a stable single-point cluster, or (2) a small number of closely packed points relatively isolated from the rest of the data. Since the random number generator is not constrained from producing such data, as verified by the data on the CRT, the detection of such closely packed clusters must be accepted as valid.

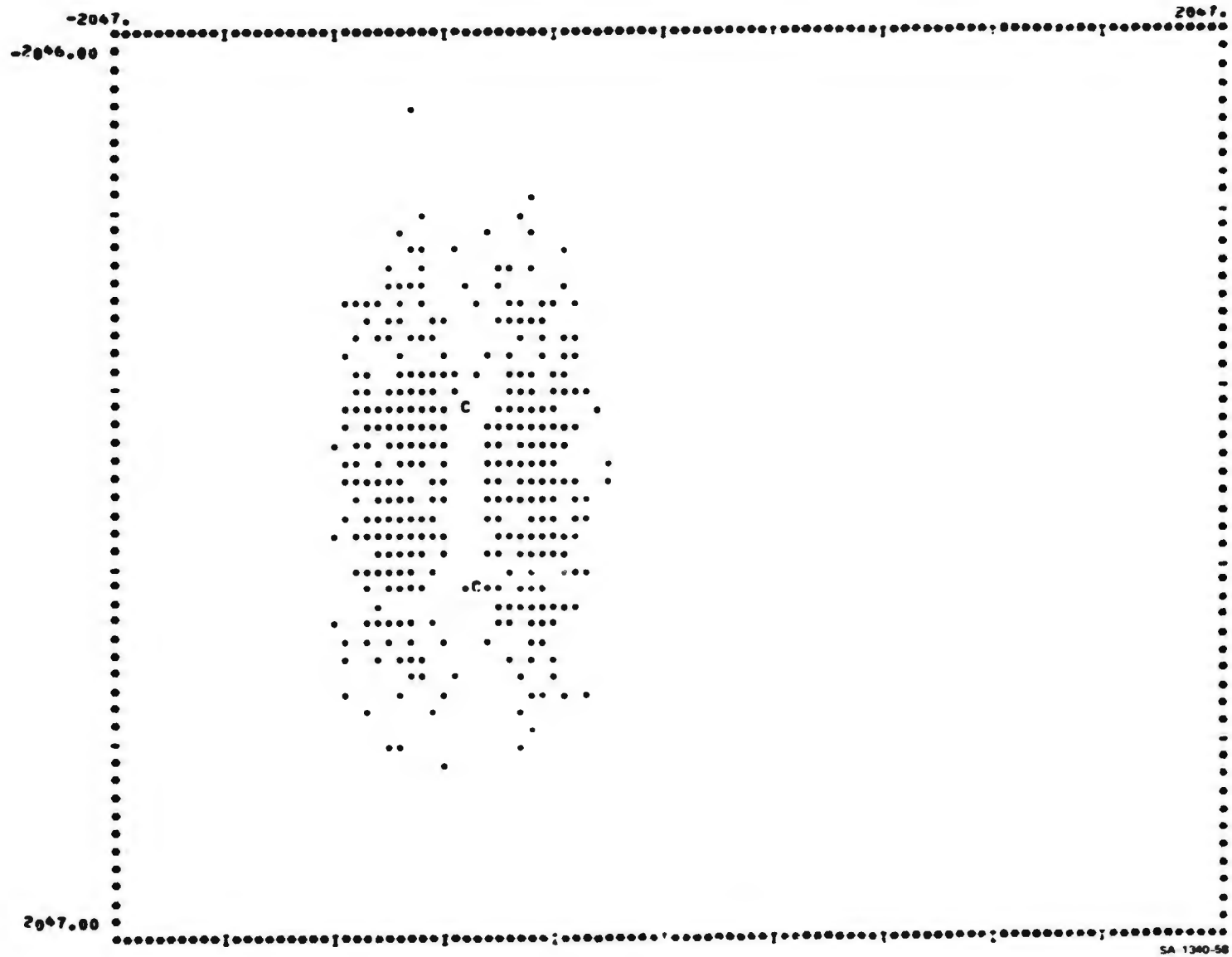


Figure 45. Unsatisfactory Experimental Clustering Using Euclidean Distance

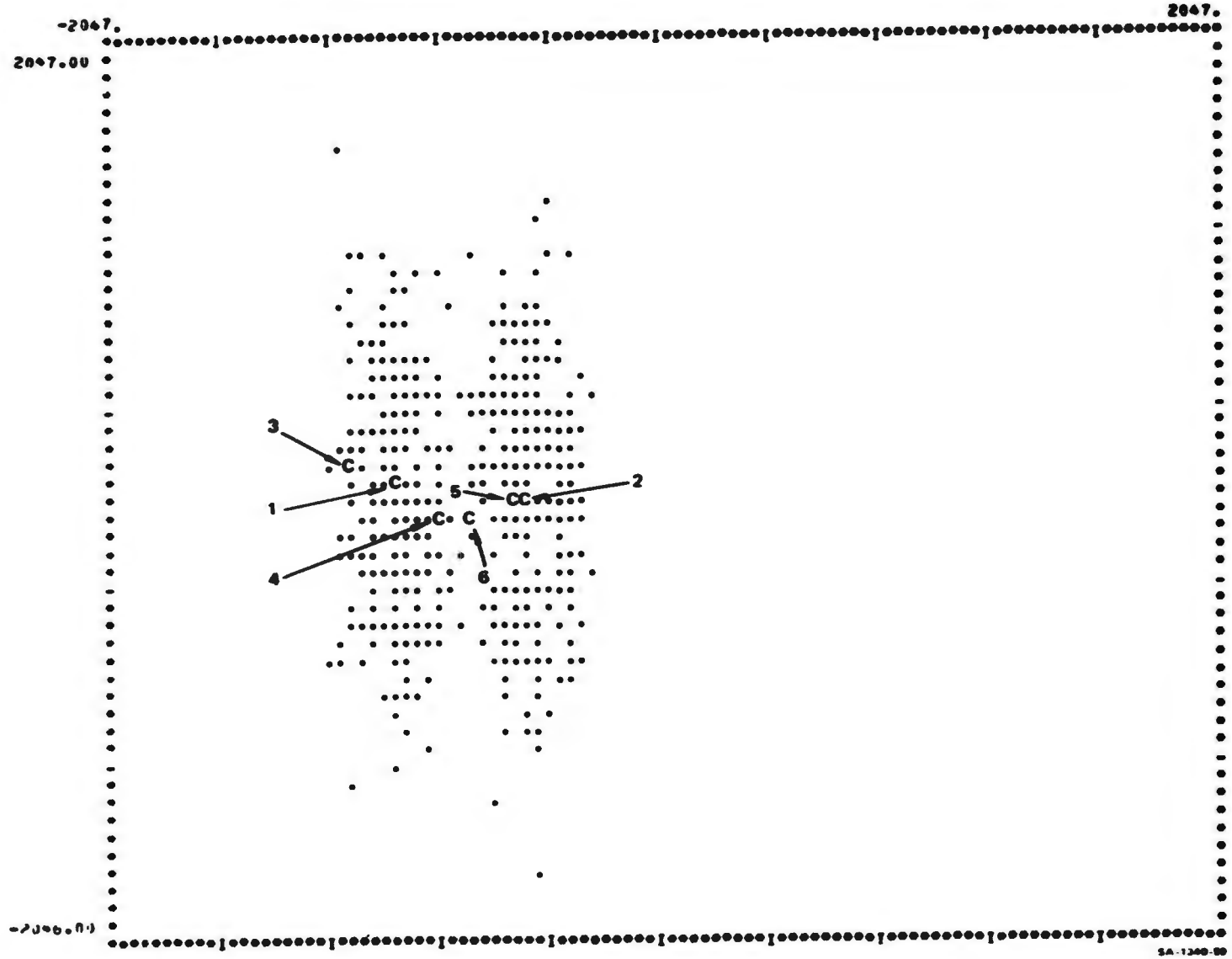


Figure 46. Satisfactory Clustering Using Generalized Distance

In the particular experiment being discussed, input and output statements in the clustering language are given in Table XIV.

TABLE XIV. SUMMARY OF EXPERIMENT ILLUSTRATED IN FIGURE 46

INPUT	N2	I1	P	300	C	-1000	0	D	100	500	I2	P	300	C	-500	0	D	100	500
GENERATED	(Different data samples were generated, although they have the same generating specifications.)																		
EUCLIDEAN	N2	I1	P	361	C	-745	-340	D	266	310	I2	P	239	C	-736	449	D	271	280
GENERALIZED	N6	I1	P	290	C	-1002	10	D	93	505	I2	P	300	C	-498	-22	D	104	530
		I3	P	6	C	-1171	93	D	39	22	I4	P	2	C	-836	-122	D	2	5
		I5	P	1	C	-560	-8	D	[a]	[a]	I6	P	1	C	-720	-102	D	[a]	[a]

[a]Only one point; so the deviation is not defined in the usual way.

Note that the input and generalized statements agree quite closely, except for the minor Clusters I3, I4, I5, and I6. These minor clusters contain only ten points altogether and represent a small error. We can thus say that (1) the input language did not specify the actual data generated because it did not specify these minor clusters , (2) the data generation procedure is in error because it did generate minor clusters, and it should have been constrained from doing this, or (3) the clustering algorithm for the program is in error because it detected these minor clusters with too high a resolution.

Our current solution to this dilemma is to increase the size of the deviation for one point in the clustering algorithm. This corresponds to decreasing the precision of the data measurements, consistent with the hypothesis that the minor clusters are an artifact and do not truly represent real subclusters in the data. Alternatively, we might require the

user to specify the data precision to be used with the generation as part of the clustering language. If the precision makes the deviation of each point larger, then the generalized distances between clusters will all typically decrease, and such minor clusters will therefore no longer remain isolated. Another way to measure the performance of the clustering program is to compare it with a perfect clustering of each and every point by means of the clustering similarity measure (CSM) as defined by Rand [139]. We have programmed this measure and used the pattern-cluster membership table of ISODATA as the input data for the CSM. For this particular clustering of the cigar data in Figure 46, the value we obtain for CSM is 0.979. A perfect clustering results in a CSM of unity.

C. A Liberal Interpretation for an Example of the Rudimentary Clustering Language

To show that the experiments are useful, we ask the reader to imagine a liberal interpretation of the presently used, rudimentary clustering language. For example, below we give an output result with its liberal interpretation alongside. (This is a decompiling function in computer science terminology--i.e., a description derived from machine-like numbers.)

Rudimentary clustering language	Liberal interpretation
I1...	There is a cluster,...
...P 100...	...having 100 points,...
...C 500 0...	...situated at coordinate 500,0,...
...D 300 300	...which has a circular shape extending about 300 coordinate units in each direction.

Compare this output description (liberal interpretation) with a very loose input description (in high-level language), as given before in Section VI-B-3. If these descriptions match, we have closed-loop recognition and generation of high-level (natural or realistic) data. By matching, we mean that the input and output descriptions must match in syntax and semantics, including vocabulary and quantitative values. In this way, we can check the accuracy of recognition by the degree of match. However, the degree of match may not have to be exact for the establishment of acceptable performance in a particular application. The term "closed-loop" is derived from the simple system block diagram shown in Figure 47, in which the presence of the closed loop is evident, and the implication of circular completeness is intended by traveling one turn around the loop. Going one turn around the loop is a universal process (uni = one, vertere = to turn). By this process, we imply that all things in this system have been considered and the system is complete, closed, or verified, when the beginning and the end states are in sufficiently close agreement.

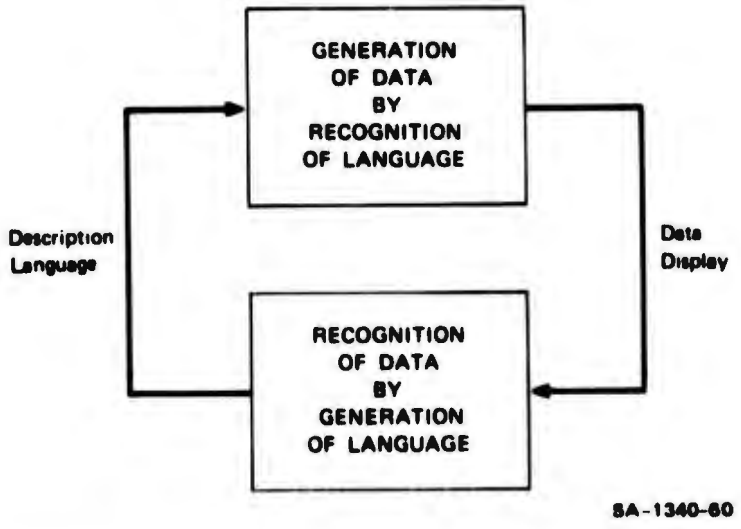


Figure 47. The Closed-Loop Generation and Recognition Methodology

The closed loop has advantages over the open loop in the research context because the beginning and end states are at the same place or state. This allows us to attribute any differences in the beginning and end states to the looping process. We can compare either the data displays or the description language at beginning and end states. In contrast, the open loop has different starting and ending places; i.e., the termination or goal is removed from the beginning, and differences may be due to either the process of getting from beginning to end or to the different states or places at the beginning and end. This does not provide a common basis for comparing results, but it is the usual mode of operation in working applications after the system has been designed and calibrated. The methodology is verified by starting at any part of the loop, traveling around the loop for one turn to reach the starting place, and comparing the starting state with the result of the process around the loop.

D. Experimental Determination of a Cluster Center by Human Judgment

The PDP 11 interactive display facility was programmed for experiments in the visual location of cluster centers by human judgments. In understanding human p.r. capability, it is important to measure parameters of human pattern perception [133]. Our clustering algorithms have generally used the center of gravity value of a set of data points as the cluster center. This is a mechanical, physical, or gravitational measure of a center. In statistics, the most common measure of central tendency is the average value, which corresponds both mathematically and conceptually to the center of gravity. (More specifically, the center of gravity is a weighted average.) However, it has been assumed that a human faced with the task of actually placing a cluster center in a set of points would choose the center of gravity or mean value. While this is probably a valid assumption for normal data, it may be very inaccurate for other

distributions [140]. Thus, the experiment we conducted and describe here tests the hypothesis that human judgments of cluster centers are not different from the center of gravity or mean value determination. We also determine what the accuracy of placement of cluster centers is, both for various individual humans and for a normal population of human viewers.

1. Experimental Procedure

The experiments began with a task description given to each viewer in the manner shown by Table VI, Computer Program for Human Experimental Subjects. The task was simply for the human to place a central point into the cloud of data such that this point was the best cluster center according to his judgment. Fifty clusters were generated using the clustering language, and they were presented on the CRT in succession to the viewer. The clusters all had the same size or deviation and were circular in the sense that deviations along the horizontal and vertical axes were specified as identical. The centers of the clusters varied about the middle of the screen; so the viewer was forced to adjust his judgment to a different position on the screen for each new presentation. The size of the cluster was small enough that the cluster could be moved around the center of the screen without coming near to the edge. Numerically, the deviation of the clusters was 300 units, and the screen size, measured in the same deviation units, was approximately 4000.

The results show that the accuracy of placement of a cluster center is typically 40 screen units (the average value for several subjects). These judgments seem surprisingly accurate when one considers that the number of points across the screen is 4096 over a physical distance of approximately 10 inches. Thus, 40 screen units represents an accuracy of $40/4096 \times 10$ inches $= 0.1$ inch. In other words, for a cluster size having a standard deviation of 300 units, the standard deviation of cluster center placement is 40 units, giving an accuracy of

40/300 x 100 = 13.3 percent. We expect the accuracy to improve as the number of points is increased, until eventually, with an infinite number of points, the accuracy of the placement of the cluster center by human observers should equal their accuracy of placement of the center in a perfect circle.

2. An Experiment to Discriminate Between One or Two Clusters

In Section IV-B, the important question of determining the number of clusters in a set of data was discussed. To investigate this question, we devised an experiment involving the presentation of either one or two clusters to the human viewer and the record of his judgment. The main parameter of interest is how large the separation between two clusters must be before the human can reliably judge that there are two clusters in the data.

The two extremes of the situation are easy to understand:

- The clusters are widely separated, so that there is a definite low density space between them. In this case, the viewer's judgment will reliably place two cluster centers in the data.
- The clusters are highly overlapping, and the mixture of two Gaussian clusters having the same mean and standard deviation values is theoretically and actually indistinguishable from one cluster having twice the number of samples as each of the individual clusters.

In this case, the viewer must decide whether to place one cluster center or two in the data. No other choice is allowed, and following well-established psychometric practice, a response must be given. The purpose of this experiment was to measure the region of uncertainty between two extremes.

After the human judgments were obtained, the same data were submitted to the ISODATA clustering algorithm to get its corresponding "judgments" and to compare human and machine algorithm results.

The experimental results of the human judgments are plotted in Figures 48 and 49. The curves are for three subjects who each participated in a one-hour experimental session. Each data point shows the pooled subjects' responses. Figure 48 shows that when the separation between cluster centers for these circular clusters of equal size was greater than 800 units, the viewer could separate the clusters correctly. As the clusters got closer, the viewer became confused and gave a smaller percentage of correct judgments. At zero distance between centers, the viewer must have been guessing (because there was no real separation); the large-sample value would theoretically be 50 percent correct. Theoretical large-sample values in the figure are indicated by a dashed curve.

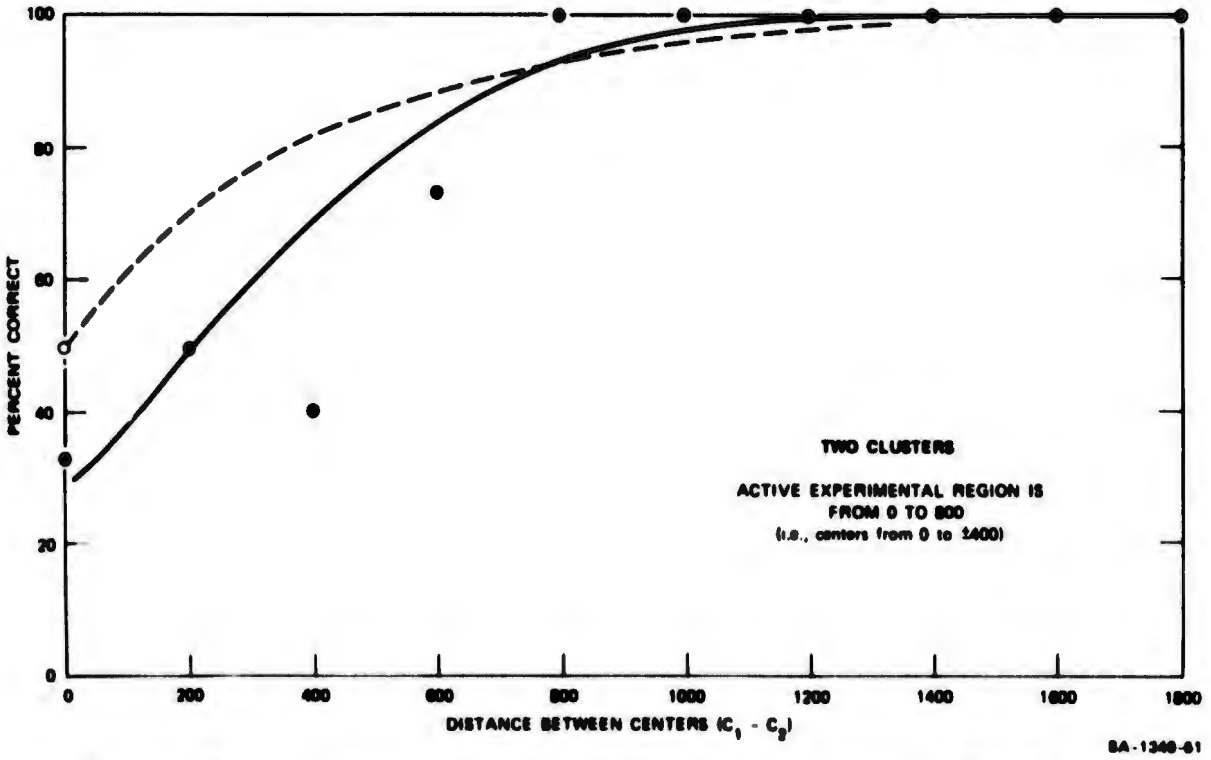


Figure 48. Experimental Results of Human Judgments of Two Clusters

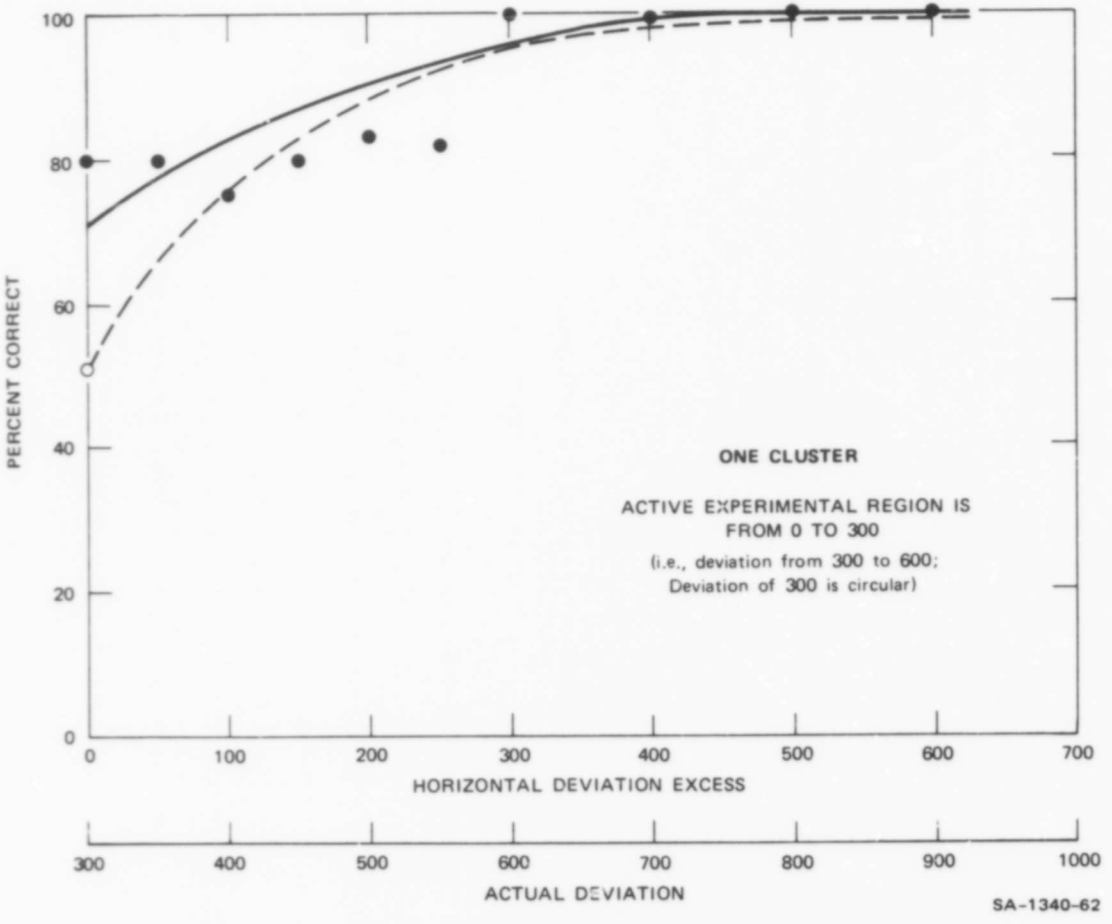


Figure 49. Experimental Results of Human Judgments of One Cluster

For the experimental results shown in Figure 49, the judgments were made only on one cluster, but the horizontal deviation was larger than the vertical deviation by a value we call the horizontal deviation excess. At zero deviation excess, the cluster is circular, and there is no physical clue suggesting the increased horizontal spread that accompanies two separated clusters. Therefore, the 50-percent correct judgment level is guesswork when the deviation excess is zero. At a deviation excess of 300 units, the viewer could reliably judge the presence of one elongated cluster, but below this value, the viewer was confused and sometimes judged a slightly elongated cluster as two separate ones.

The clustering algorithm is generally programmed to explore a range of clusterings, starting with one cluster and then splitting by stages up to eight clusters, and then lumping back down to one cluster again. This range of clusterings enables the choice of the most suitable clustering for the data. In this case, however, we were interested only in deciding between one or two clusters. We made use of the total squared error (TSE) statistic for this purpose [87].

The TSE for one cluster was normalized to be 100 percent, and we inspected the value for two clusters to test the validity of the two-cluster clustering. Whether there were actually two clusters or one in the data, the TSE was approximately 68 percent at zero cluster spacing or zero excess deviation. If the cluster spacing for two clusters or the excess deviation for one cluster was larger, the TSE dropped. However, there was more difference between these two sets of TSE values as the cluster spacing and excess deviation became larger. At a cluster spacing of 1800 units, a perfect clustering for two clusters was obtained, but at closer spacings, some data points always overlapped. At this 1800 spacing, the TSE was 17.9, with zero clustering error; so the spacing distance found by the program was the true spacing. At closer spacings than this, the program gave a spacing that was less than the true spacing

because of the overlap. However, the two clusters in this experiment were also sometimes very close together, and at some critical spacing, the program's spacing became greater than the actual spacing because the clusters were overlapping considerably.

We propose the value of TSE at this crossover point as the threshold below which the program can confidently give a two-cluster judgment. In our experiments, this critical value was TSE = 38.9 percent. Above this value, there was a no-decision region in which even a human viewer could not judge correctly, as shown by Figures 48 and 49. At very high values of the statistic TSE, we can be confident that the two-cluster TSE is invalid for two distinct clusters, thus giving clear validation of one cluster in the data. From our experiments, we chose a value of TSE = 50.0 percent for this threshold. Note that this is also the theoretical value of the TSE derived mathematically for two clusters in uniform random data [87]. We are thus conservative in selecting a threshold at a value that makes any TSE above 50.0 percent indicate one cluster. For a TSE between 50.0 and 38.9 percent, no confident judgment can be given. This quite accurately simulates the human viewing situation.

E. The Suitability of Cloud Data for Clustering Research

1. Introduction

During the course of this project, we were concurrently studying the meteorological problem of determining wind motions from satellite cloud photographs [141, 142]. Since cloud motions are often produced by winds, we were interested in determining the motions of clouds as approximations to the winds. However, a cloud is often ephemeral and unstable, and it may change its shape or disappear completely. Therefore, the identification and tracking of clouds is a problem that requires a descriptive approach subtle enough to match the ephemeral character of the

data. Thus, the a priori specification of cloud types and labels for them is not a viable approach. However, temporary recognition of particular clouds is necessary to track them in successive pictures. This tracking is usually done by displaying successive picture frames in quick succession to a human. Remarkable visual abilities, not easily specified so that they can be programmed, allow the human to track cloud motions on a cloud console [143] or film loop.

In the computer processing of successive satellite pictures, we seek machine descriptions of clouds for which no prior labels exist. However, some cloud features do persist, and we are able to generate statistical descriptors or labels for these in the clustering program. This is an ideal statement of the descriptive clustering problem, and it is one reason why we chose to investigate these cloud data. These data also have an interesting relationship to the data we have generated artificially; clouds of data were generated artificially and data from clouds were obtained from satellite measurements. Furthermore, the meteorological problem is of direct interest to the Air Force, as specified in the Statement of Work in this contract.

We have processed satellite cloud data both with and without using generalized distance, and the results are different, to some extent. However, the significance of these differences in this application is difficult to evaluate because of the richness of the real data. The results obtained in both cases are meteorologically acceptable, and because there is no independent crosscheck, such as rawinsonde wind readings available for the same area at the same time, an evaluation of the algorithm is almost impossible. The best evaluation might be obtained through use of the SRI cloud console in viewing the satellite pictures in time-lapse motion. In this case, the correlation between (1) the clustering results and the wind-motion program and (2) the very complicated photographic scene in dynamic motion on the console must be done by a human making

subjective judgments. These judgments are not easy to make because the comparison is between data of different formats. Thus, fine differences between a method that works well (Euclidean distance ISODATA) and a method that probably works better (generalized distance ISODATA) are not easy to evaluate with real data from a complex application process. This emphasizes the importance of using artificial data whose characteristics are known for validating a clustering or recognition algorithm. Our aim is to work continually with artificial data that is increasingly realistic. This also implies that our descriptions or clusterings of realistic data must be increasingly realistic and meteorologically accurate.

We next discuss the more realistic boundaries produced by using generalized distance rather than Euclidean distance in our clustering algorithms.

2. Generalized Distance Applied to Clustering Clouds from Satellite Photographs

The generalized distance produces a clustering result that is physically feasible from a meteorological point of view, whereas Euclidean distance produces contours of the cluster regions that are linear and discontinuous and that meet at sharp angles. The visual impression of boundaries from a satellite photograph or weather map is not compatible with Euclidean boundaries, whereas generalized distance clustering results in a set of contours that might have actual meteorological significance. In the studies so far, we have considered these boundaries only as isobrightness contours, but we conjecture that there may be an interesting relationship between brightness and some related atmospheric quantity that could be displayed by generalized distance boundaries. It is, however, safely beyond conjecture to say that Euclidean, perpendicular-bisector boundaries are not natural for clouds or meteorological contours in general.

F. The Use of Secondary Criteria in Lumping Clusters

Our current ISODATA lumping algorithm is based upon the distance between the two cluster centers being considered for lumping together. However, distance, or difference in spatial position, is not the only measure of relationship we could apply in assigning clusters (i.e., lumping) or data points to the same cluster membership.

This raises the basic question of whether we can have two different cluster centers existing at almost the same point in space without lumping them. The answer is that if distance (even generalized distance) is the only criterion for lumping, then two cluster points cannot simultaneously exist at one position in space.

However, it is useful to be able to represent two different types of clusters that have, by chance or design, their centers at similar positions in space. We are referring to annular data in which a large ring of data circumscribes a smaller cluster inside, as sketched in Figure 30(c). (Zahn [52] has given examples of these.) To avoid lumping dissimilar clusters that are close together in space, some secondary characteristic must be used, such as their standard deviations or sizes along each axis. Because we need to consider multivariate clusters, we may also use a more general size descriptor, such as the rms distance within a cluster, which is routinely calculated in the ISODATA program. Other secondary characteristics are number of points in the cluster, and third and fourth moments of the distribution (skewness and kurtosis).

We already use this principle in our cloud clustering work [142]. In the MOTION program, pairs of cluster centers are not linked or joined by a wind motion vector unless the (Euclidean) distance between the two centers is reasonably small and the secondary characteristics or descriptions of each of the clusters are nearly compatible (i.e., the cloud in the second picture must be recognized as corresponding to a certain cloud

in the first picture). This linking of cluster centers is a type of lumping in the time domain for successive cloud pictures that are taken 25 minutes apart.

To cluster annular data sets (which are usually cited as difficult to handle) correctly, or to cluster even more difficult data, we must introduce the secondary characteristics into the lumping criteria. The pairing association criterion used in the MOTION program is similar to the lumping criterion and provides a practical illustration for the usefulness of secondary characteristics in providing a more meaningful description [144].

Note that the ultimate description in terms of accuracy, if we rely on measurement (rather than on an interpretation or meaning of the measurement), is the restatement of the input data themselves. This is usually most compactly done in the form of a table. However, such a table is not a suitable means of input to a human; some form of graphical display is usually a more meaningful interpretation to a human. In other words, we might say that the measurements are suited to or relevant for input to the computer and not to the human. The function of the computer in this clustering application is to transform the raw measurement data into a suitable description. This description may be in verbal form with summary statistics, as in the output from the improved ISODATA program, or it may be in graphical form, such as that obtained by feeding the output language into the data displaying programs to give a visual or graphical [145] description. In computer science terms, the clustering program transforms measurements, examples, or instances of data and produces a suitable general data structure [146] as the result of the computation. This data structure can then be used to structure a file, or it can itself be a compressed version of the full file of data, from which a directory or retrieval scheme may be derived.

The combination of these secondary descriptive characteristics of clusters into a more suitable measure of relationship is crucial to achieving a correct clustering in practical, nontrivial applications. We believe that the complex perceptual problems of tracking clouds of digital data representing real clouds can best be approached by using these new measures of relationship and a more general concept of distance, in accordance with the physical principles of relativity, and the application of these ideas to problems of influence and relationship in pattern recognition.

SECTION VIII

DESCRIPTION OF COMPUTER PROGRAMS AND FACILITIES

We feel that the reader will be interested in the computer environment for this work. Three different computers having various advantages and disadvantages were used.

A. The PDP 10 Facility

The computing work on the minimal spanning tree and the Kolmogorov-Smirnov test used a PDP 10 computer. The PDP 10 is maintained and operated by the Artificial Intelligence Center at SRI, and it provides extensive time-sharing facilities to many users. An ADAGE CRT display was used for some of this work. These computations were programmed in FORTRAN.

B. The CDC 6400 and PDP 11 Facilities

The computing experiments for the development of a methodology for closed-loop generation and recognition were performed on a CDC 6400 and a PDP 11. The CDC 6400 is the central computing facility at SRI, and the PDP 11 is administered by the Information Science Laboratory. Table XV summarizes some of the features of these facilities. Note that the magnetic tape units are not compatible. The interface and common features are the:

- Clustering language
- Random and Gaussian generators
- FORTRAN compilers.

TABLE XV. SUMMARY OF THE FEATURES OF THE CDC 6400 AND THE PDP 11

CDC 6400	PDP 11
ISODATA algorithm	Human judgments
Printer plots	CRT display and printer plots
Batch operation	Interactive operation
Input media	Input media
Cards	Cards
Magnetic tape (seven track for cloud data)	Teletype
	Mouse
	Keyboard on display console
	Magnetic tape (nine track and DECTAPE)
Output media	Output media
Cards	CRT
Printout	Teletype
Magnetic tape (seven track)	Printout
	Magnetic tape (nine track and DECTAPE)

The ISODATA clustering program has been improved as a result of this contract in the following ways:

- Introduction of Mahalanobis, or generalized, distance.
- Addition of a descriptive output language suitable for both man and machine.
- Addition of line printer plot outputs and improved tabular output arrangements that aid in program diagnosis and data interpretation.
- Acceptance of a descriptive input language (the same as that for output) that generates data for clustering and display. This display is either in CRT form (on the PDP 11) or line printer form (on both CDC 6400 and PDP 11).

Other computer programs that were needed to support the main line of investigation are:

- Random number generators. Identical versions had to be derived for both the CDC 6400 and the PDP 11 because no suitable common data transfer medium was available that was compatible with the mode of interactive operation.
- A character code converter to convert from the keypunch code of the PDP 11 to the different keypunch code on the CDC 6400. This allowed us to compile FORTRAN programs written for the PDP 11 on the CDC 6400 which was a considerable convenience due to the superior service and incidental access afforded to the FORTRAN compiler by the batch operation of the CDC 6400 service.

SECTION IX

SUMMARY AND CONCLUSIONS

This section summarizes the results of each section, draws conclusions or makes deductions from these results, and makes recommendations based on these results.

A. Section I

Section I declares the objectives of the study and the scope of each section of the report. Each section is related to the set of study objectives. The field of p.r. is defined very broadly and is related to clustering, statistics, artificial intelligence, perception psychology, and interactive graphic computation. The importance of a linguistic or descriptive approach is stressed, based on the nature of programming languages and the need to program recognition algorithms.

B. Sections II and III

Section II provides an overview of the application of statistics to p.r. Of the many varied applications, one common characteristic is the difficulty of working with multidimensional, nonnormal, poorly understood distribution functions. The frequent failure of methods based on a multivariate normal model, and the computational hopelessness of completely general nonparametric techniques, have spurred the development of clustering techniques. These techniques, surveyed in some detail, represent an attempt to find data models that are more realistic than the multivariate normal model while retaining reasonable computational requirements.

From a practical viewpoint, more experimental work with both real and artificial data is needed to establish the empirical validity of these models and to shape them to better fit the structure of real data. Section IV illustrates the advantages of interactive systems for such

Preceding page blank

work. From a theoretical viewpoint, a number of topics deserve attention. To study optimality, alternative formulations of the clustering problem should be considered to see if they lead to more tractable analytical problems. The question of the validity of cluster descriptions needs more attention, as do some basic questions on the decomposition of mixture densities. However, such work faces the danger of becoming arid if it loses contact with its source in practical applications.

Section III provides an overview of some of the applications that have been made of p.r. methodologies. While a great many applications have been reported in the literature, most of these have been study efforts that merely illustrated the potential applicability of some technique. Relatively few have led to useful commercial or military systems. In some cases, technical solutions are available, but the cost of a system that meets technical requirements is excessive. More often, not enough is known about the various aspects of a particular application to allow a system to be designed with adequate confidence of successful results. Finally, there are situations in which none of the known techniques have produced satisfactory results, and further progress awaits the development of new methods for p.r.

C. Section IV

Section IV discusses the development of new methods. Several diverse topics are treated as new methods, but all have to deal with improved clustering or description of data.

Some basic questions concerning a data set to be clustered are:

- How many cluster centers should be used to describe it?
- What is the quality or effectiveness of this description?
- How valid or unique is it?

The well-documented Kolmogorov-Smirnov test has been applied to the question of cluster validity or quality. In this study, a multivariate extension of the test was developed, and Monte Carlo experiments were run to determine critical values as a function of sample size and dimensionality.

The graph-theoretic or link-node approaches to clustering, also called minimal spanning tree methods, are often costly to apply because of their extensive computation time. Section IV presents a new approximate method of computing these trees. If n is the number of nodes in the tree, the required computation time for the new method grows as $n\sqrt{n}$ rather than as n^2, and empirical tests showed the approximate method to be computationally advantageous for $n \geq 32$.

D. Section V

Traditionally, Euclidean distance has been used as the measure of relationship between data points in clustering programs. Section V gives reasons why this measure is not adequate for some significant problems of simple gestalt perception and description of clusters. Because a refinement in the geometric concept of relationship is required, it is natural to consider relativity in a manner analogous to the considerations of Einstein in the realm of physics.

The basic notion of clustering is to find a center representative of several other points. This representative center has a greater domain or region of coverage of the space than does each of the data points. However, each data sample (i.e., point) itself represents a region of the data space, and it is not dimensionless. These fundamental concepts bear upon the basic issues of clustering and lead to a concept of generalized (or relativistic) distance that is a more suitable measure of relationship than Euclidean distance for perceptual problems of gestalt recognition.

Before reporting how generalized distance was used in a series of software experiments, Section V discusses various methods of determining centers and boundaries and provides also a unifying viewpoint for clustering problems.

Since compactness is an essential property of a well-clustered data set, and since graph-theoretic techniques for clustering are promising, a derivation of a graph-theoretic measure of compactness is presented.

E. Section VI

To experiment effectively with the generalized distance measure, and for several other reasons discussed more fully in Section VI, we devised an interactive display program to gain some of the advantages of man/machine computation. We feel that the benefits of this interactive graphic computation are not generally appreciated in the field of p.r. and clustering research. Therefore, Section VI expounds this view.

The above considerations lead logically to the recognition facility that we have developed for the closed-loop generation and recognition of data, and to the development of a computer language for clustering. This language is only rudimentary because we wish primarily to establish the methodology or principles underlying its development and because development of a more extensive language is beyond the resources of this project.

Basically, two sets of experiments were carried out in the closed-loop methodology, although they are related. In the machine algorithm experiments, we used batch processing on the CDC 6400 computer. In the analogous experiments on human clustering of data on a CRT display, we used the PDP 11 computer. Because the same clustering language facilities were developed for both computers, we were able to generate and to perform recognition of identical data on both machines, despite the lack of a hardware connection between them. The main purposes of our experiments were as follows:

- To demonstrate the methodology of model building and iterative evolution (as discussed on page 6 of the proposal). See Section VI-A-1.
- To demonstrate the use of a computer language for the quantitative description of clusters and for their generation. See Section VI-B.
- To use the demonstration facilities to attack some of the basic problems in clustering, particularly the following questions.
  - How many clusters should be used to describe the data?
  - What type of center do humans use visually, how accurate is it, and how does it compare to the algorithmic computations of the clustering program?
  - What improvements can be made to the clustering algorithm by using the generalized distance measure of relationship?

Section VI-A discusses the evolutionary development of new methods through the use of a man/machine facility. Basically, our contention is that the interactive process is more sensitive because of the increased number of turnarounds, the user's greater control over the program, and the incremental nature of the computing activity. This is especially evident when there is good graphic information displayed to the user by means of a CRT display. We were able to put this methodology into practice using the Vector General display of the PDP 11 computer. The scope of the project did not allow us to progress to the use of very complex data models, but we were able to generate a wide variety of Gaussian clusters by means of a simple clustering language specification. Several examples of the use of the clustering language are given in Section VI-B.

F. Section VII

Section VII illustrates the use of generalized distance on some data that are difficult to cluster using Euclidean distance. A new lumping algorithm, necessitated by the different way in which generalized distance operates, was developed.

In an experiment using human visual judgments, we determined the accuracy of location of cluster centers compared to the computed average values for those centers. In another visual experiment, the viewer was required to judge whether the data consisted of one or two clusters as the separation between clusters was increased in random sequential presentations. Some of these presentations actually consisted of only one cluster; so the correct answer had to be based on the evidence, according to established psychometric testing methods. The one-cluster presentations in this experiment were of clusters elongated in a manner similar to the two-cluster presentations along the same axis of separation as that for the two-cluster presentations. We thus measured the region of uncertainty, or the probability of error in human judgment, related to the number of clusters. This attacked the basic problem of the number of clusters to use, and it is related to the discussion of the Kolmogonov-Smirnov test given in Section IV-B-2.

The same data used in the human experiments was clustered by the ISODATA program, and a test statistic, discussed more fully in Section IV, was provided by the program to decide on the number of clusters. This test statistic approximately agreed with the human judgments.

G. Section VIII

Section VIII briefly mentions some of the computing facilities used and lists some of the computing features used and some of the computer software developed.

H. Recommendations

In general, we recommend that this work be continued and extended, particularly in the use of more complex data, leading from the artificial to real situations in a complete and continuous evolutionary process.

Basic research on recognition methods, such as the research conducted on this project, is contributing generally to the needs of the government and to the technological services of the nation. We also recommend that the Air Force be more specific about its recognition problems if specific solutions are desired. We feel that these problems can be related to the overall purposes of the Air Force in a specific and logical way.

We recommend a linguistic or descriptive approach to data analysis that includes p.r. and clustering. Contextual factors play such an important role in the recognition function that to ignore them makes the task of recognition much more difficult. This leads us to recommend the further development of computer languages for recognition and clustering that we began on this project. In addition, the development of a language for the specification of recognition tasks or systems would be valuable. We speculate that a particular p.r. task specified in this language might eventually be compiled by a general p.r. problem-solving system to provide a solution for the p.r. task semiautomatically (with human assistance in man/machine interaction).

Further work needs to be done in developing and testing simpler statistics for cluster validity and in relating these to human judgments. The MST methods could be more widely applied and further investigated as to their combination with other clustering methods.

Appendix

LISTING OF FORTRAN PROGRAMS FOR RECOGNITION EXPERIMENTS

The listings for the following programs are presented in this appendix:

- Subroutine KS--referred to in Section IV-B.
- Subroutines MSTREE and APMST--referred to in Section IV-C.
- Program ISODATA and Subroutines CLUST and LUMP--referred to in Section VII-A-2. These subroutines are part of the ISODATA program, which is too lengthy to list here. A copy of it can be obtained from the Defense Documentation Center under the number AD694114.
- Program PLACE for the PDP 11--referred to in Section VI-B.

Listing for Subroutine KS

```
      SUBROUTINE KS(N,ND,X,DN,XN)
C
C*THE KOLMOGOROV-SMIRNOV ONE-SAMPLE STATISTIC*
C*NULL HYPOTHESIS: INDEPENDENT NORMAL DISTRIBUTION*
C*EXHAUSTIVE VERSION*
C
C       N  = SAMPLE SIZE
C       ND = DIMENSIONALITY, .LE. 100
C       X  = DATA MATRIX, ND BY N
C       DN = K-S STATISTIC
C       XN = ND-ELEMENT ARRAY FOR POINT WHERE MAXIMUM IS ACHIEVED
C
      DIMENSION X(ND,N),XN(ND),XMEAN(100),XSIGMA(100),XK(100),
     1 KNT(100)
      RECIP=1./N
C
C*COMPUTE SAMPLE MEANS AND STANDARD DEVIATIONS*
C
      DO 14 ID=1,ND
      XM=0.
      DO 12 K=1,N
12    XM=XM+X(ID,K)
14    XMEAN(ID)=RECIP*XM
C
      DO 18 ID=1,ND
      XSIG=0.
      DO 16 K=1,N
      E=X(ID,K)-XMEAN(ID)
16    XSIG=XSIG+E*E
18    XSIGMA(ID)=SQRT(RECIP*XSIG)
C
C*INITIALIZE*
C
      DN=0.
      DO 19 ID=1,ND
19    KNT(ID)=1
      KNT(1)=0
C
C*UPDATE COUNTERS*
C
25    ID=1
26    KNT(ID)=KNT(ID)+1
      IF(KNT(ID).LE.N)GO TO 28
      IF(ID.GE.ND)RETURN
      KNT(ID)=1
      ID=ID+1
      GO TO 26
28    DO 29 ID=1,ND
29    XK(ID)=X(ID,KNT(ID))
C
C*COMPUTE CDF*
C
```

```
          KOUNT=0
          KTIE=0
          DO 35 KK=1,N
          ITIE=.FALSE.
          DO 33 ID=1,ND
          E=X(ID,KK)-XK(ID)
          IF(E.GT.0.)GO TO 35
          IF(E.EQ.0.)ITIE=.TRUE.
33        CONTINUE
          KOUNT=KOUNT+1
          IF(ITIE)KTIE=KTIE+1
35        CONTINUE
          IF(KTIE.LE.0)GO TO 25
C
C*COMPUTE CDF*
C
          CDF=1.
          DO 42 ID=1,ND
          E=XK(ID)-XMEAN(ID)
          IF(XSIGMA(ID).GT.0.1*ABS(E))GO TO 41
          IF(E.GT.0.)E=1.
          IF(E.EQ.0.)E=0.5
          IF(E.LT.0.)E=0.
          GO TO 42
41        E=ERF(E/XSIGMA(ID))
42        CDF=CDF*E
C
C*UPDATE DEVIATION*
C
          ECDFM=RECIP*(KOUNT-KTIE)
          ECDFP=RECIP*KOUNT
          EMINUS=CDF-ECDFM
          IF(DN.GE.EMINUS)GO TO 45
          DN=EMINUS
          DO 44 ID=1,ND
44        XN(ID)=XK(ID)
45        EPLUS=ECDFP-CDF
          IF(DN.GE.EPLUS)GO TO 25
          DN=EPLUS
          DO 46 ID=1,ND
46        XN(ID)=XK(ID)
          GO TO 25
          END
```

Listings for Subroutines MSTREE and APMST

```
      SUBROUTINE MSTREE(NX,ND,X,N,NIT,MST1,MST2,UI,JI)
C*SUBROUTINE FOR FINDING THE MINIMAL SPANNING TREE*
C       NX = NUMBER OF POINTS
C       ND = DIMENSIONALITY
C        X = DATA MATRIX, ND BY NX
C        N = NUMBER OF NODES, N.LE.NX
C       NIT= N-ELEMENT ARRAY OF NODES TO BE IN THE TREE
C       MST1 AND MST2: N-ELEMENT MATRICES
C       UI = N-ELEMENT SCRATCH ARRAY
C       JI = N-ELEMENT SCRATCH ARRAY
C
      DIMENSION X(ND,NX),NIT(N),MST1(N),MST2(N),UI(N),JI(N)
C
C*INITIALIZE NODE LABEL ARRAYS*
C
      IMST=0
      KP=NIT(N)
      NITP=N-1
      DO 100 I=1,NITP
      NI=NIT(I)
      D2=0.
      DO 50 ID=1,ND
50    D2=D2+(X(ID,NI)-X(ID,KP))**2
      UI(I)=D2
100   JI(I)=KP
      GO TO 350
C
C*UPDATE LABELS OF NODES NOT YET IN TREE*
C
200   DO 300 I=1,NITP
      NI=NIT(I)
      D2=0.
      DO 250 ID=1,ND
250   D2=D2+(X(ID,NI)-X(ID,KP))**2
      IF(UI(I).LE.D2)GO TO 300
      UI(I)=D2
      JI(I)=KP
300   CONTINUE
C
C*FIND NODE NOT IN TREE NEAREST TO TREE*
C
350   UK=UI(1)
      K=1
      DO 400 I=1,NITP
      IF(UI(I).GE.UK)GO TO 400
      UK=UI(I)
      K=I
400   CONTINUE
C
C*SAVE NODES OF NEW EDGE*
C
      IMST=IMST+1
      MST1(IMST)=NIT(K)
      MST2(IMST)=JI(K)
      KP=NIT(K)
C
C*DELETE NEW NODE FROM ARRAY*
C
```

```
          UI(K)=UI(NITP)
          NIT(K)=NIT(NITP)
          JI(K)=JI(NITP)
          NITP=NITP-1
C
C*WHEN ALL NODES ARE IN TREE, QUIT*
C
          IF(NITP.GT.0)GO TO 200
          RETURN
          END
```

```
      SUBROUTINE APMST(N,ND,NGMAX,X,KX,XCG,MST1,MST2)
C
C*ROUTINE TO FIND APPROXIMATE MINIMAL SPANNING TREE*
C       N = NUMBER OF POINTS
C       ND = DIMENSIONALITY
C       NGMAX = MAX. NO. GROUPS, TYPICALLY SQRT(N)
C       X = DATA MATRIX, ND BY N
C       KX = N-ELEMENT POINTER MATRIX
C       XCG = CENTER OF GRAVITY MATRIX, ND BY NGMAX
C       MST1 AND MST2: THE N-ELEMENT ANSWER MATRICES
C
C       NCG(IG) = NUMBER OF POINTS IN GROUP IG
C       KIX(IG) = FIRST POINT IN GROUP IG
C       LIX(IG) = LAST POINT IN GROUP IG
C       REMAINING ELEMENTS ACCESSED THROUGH KX
C
      DIMENSION X(ND,N),KX(N),XCG(ND,NGMAX),MST1(N),MST2(N)
      DIMENSION NCG(100),KIX(100),LIX(100),W(100),
     1 KPTR(100),MSTI(100),MSTJ(100),UI(100),JI(100)
      DATA HUGE/1.E38/
C
C
C*FORM NG GROUPS*
C
C
C*INITIALIZE*
C
      NG=2
      DO 15 IG=1,NG
      NCG(IG)=1
      KIX(IG)=IG
      LIX(IG)=IG
      DO 15 ID=1,ND
15    XCG(ID,IG)=X(ID,IG)
      NGSQRD=NG*NG
      K=2
C
C*MAIN LOOP*
C
20    K=K+1
      IF(K.GT.N)GO TO 156
C
C*FIND NEAREST MEAN*
C
      DMIN2=HUGE
      DO 35 IG=1,NG
      D2=0.
      DO 32 ID=1,ND
32    D2=D2+(X(ID,K)-XCG(ID,IG))**2
      IF(D2.GE.DMIN2)GO TO 35
      DMIN2=D2
      IGMIN=IG
35    CONTINUE
C
C*UPDATE*
C
      IG=IGMIN
      KX(LIX(IG))=K
```

```
        LIX(IG)=K
        NCG(IG)=NCG(IG)+1
        FACTOR=1./NCG(IG)
        DO 45 ID=1,ND
45      XCG(ID,IG)=XCG(ID,IG)+FACTOR*(X(ID,K)-XCG(ID,IG))
C
C*TEST FOR SPLITTING*
C
        IF(K.LE.NGSQRD)GO TO 20
C
C*FIND LARGEST CLUSTER*
C
        NMAX=0
        DO 55 IG=1,NG
        IF(NCG(IG).LE.NMAX)GO TO 55
        NMAX=NCG(IG)
        IGMAX=IG
55      CONTINUE
        IG=IGMAX
C
C*FIND POINT FARTHEST FROM MEAN*
C
        DMAX2=0.
        KK=KIX(IG)
        KL=LIX(IG)
        GO TO 62
61      KK=KX(KK)
62      D2=0.
        DO 64 ID=1,ND
64      D2=D2+(X(ID,KK)-XCG(ID,IG))**2
        IF(D2.LE.DMAX2)GO TO 66
        DMAX2=D2
        KKMAX=KK
66      IF(KK.NE.KL)GO TO 61
C
C*FIND WEIGHT VECTOR*
C
        DO 72 ID=1,ND
72      W(ID)=X(ID,KKMAX)-XCG(ID,IG)
        W0=0.
        DO 74 ID=1,ND
74      W0=W0-W(ID)*XCG(ID,IG)
C
C*SPLIT OFF NEW CLUSTER*
C
        NG=NG+1
        NGSQRD=NG*NG
        KK=KIX(IG)
        KL=LIX(IG)
        NG1=0
        NG2=0
        IF1=.TRUE.
        IF2=.TRUE.
        DO 80 ID=1,ND
        XCG(ID,IG)=0.
80      XCG(ID,NG)=0.
        GO TO 82
81      KK=KX(KK)
```

```
82      DOT=W0
        DO 84 ID=1,ND
84      DOT=DOT+W(ID)*X(ID,KK)
        IF(DOT.LE.0.)GO TO 100
C
C*ASSIGN TO CLASS IG*
C
        NG1=NG1+1
        FACTOR=1./NG1
        DO 92 ID=1,ND
92      XCG(ID,IG)=XCG(ID,IG)+FACTOR*(X(ID,KK)-XCG(ID,IG))
        IF(IF1)GO TO 95
        KX(K1)=KK
        K1=KK
        GO TO 110
95      IF1=.FALSE.
        KIX(IG)=KK
        K1=KK
        GO TO 110
C
C*ASSIGN TO CLASS NG*
C
100     NG2=NG2+1
        FACTOR=1./NG2
        DO 102 ID=1,ND
102     XCG(ID,NG)=XCG(ID,NG)+FACTOR*(X(ID,KK)-XCG(ID,NG))
        IF(IF2)GO TO 105
        KX(K2)=KK
        K2=KK
        GO TO 110
105     IF2=.FALSE.
        KIX(NG)=KK
        K2=KK
C
C*END OF SPLIT LOOP*
C
110     IF(KK.NE.KL)GO TO 81
        LIX(IG)=K1
        LIX(NG)=K2
        NCG(IG)=NG1
        NCG(NG)=NG2
        GO TO 20
```

```
C
C
C*FIND AND LINK MST'S*
C
C
C*FIND MST'S FOR NG GROUPS*
C
150     NODE=1
        DO 158 IG=1,NG
        IF(NCG(IG).LE.1)GO TO 158
        I=1
        KPTR(1)=KIX(IG)
        KL=LIX(IG)
152     I=I+1
        KPTR(I)=KX(KPTR(I-1))
155     IF(KPTR(I).NE.KL)GO TO 152
        CALL MSTREE(N,ND,X,NCG(IG),KPTR,MST1(NODE),MST2(NODE),
        1 UI,JI)
158     NODE=NODE+NCG(IG)-1
        LASTNODE=NODE-1
C
C*FIND MST FOR MEANS*
C
        DO 162 I=1,NG
162     KPTR(I)=I
        CALL MSTREE(NG,ND,XCG,NG,KPTR,MSTI,MSTJ,UI,JI)
C
C*LOOP THROUGH MEAN MST EDGES*
C
        DO 200 K=1,NG-1
        IG=MSTI(K)
        JG=MSTJ(K)
        IF(NCG(IG).GE.NCG(JG))GO TO 187
        KG=IG
        IG=JG
        JG=KG
C
C*FIND POINT IN GROUP I NEAREST TO MEAN J*
C
180     DMIN2=HUGE
        KK=KIX(IG)
        KL=LIX(IG)
        GO TO 182
181     KK=KX(KK)
182     D2=0.
        DO 184 ID=1,ND
184     D2=D2+(X(ID,KK)-XCG(ID,JG))**2
        IF(D2.GE.DMIN2)GO TO 186
        DMIN2=D2
        KKMIN=KK
186     IF(KK.NE.KL)GO TO 181
        KI=KKMIN
C
C*FIND NEAREST POINT IN GROUP J*
C
        DMIN2=HUGE
        KK=KIX(JG)
        KL=LIX(JG)
```

```
          GO TO 192
191       KK=KX(KK)
192       D2=0.
          DO 194 ID=1,ND
194       D2=D2+(X(ID,KK)-X(ID,KI))**2
          IF(D2.GE.DMIN2)GO TO 196
          DMIN2=D2
          KKMIN=KK
196       IF(KK.NE.KL)GO TO 191
          KJ=KKMIN
C
C*LINK MST-I AND MST-J*
C
          MST1(NODE)=KI
          MST2(NODE)=KJ
200       NODE=NODE+1
          KX(1)=LASTNODE
          RETURN
          END
```

Listing for Program ISODATA

and Subroutines CLUST and LUMP

```
      PROGRAM ISODATA(INPUT,OUTPUT,PUNCH=130
     *          ,TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=PUNCH,TAPE20=130)
    1 FORMAT (*1PROGRAM ISODATA, VERSION 20 NOV 72*)
      COMMON /NOV20/  XXXXXX
C  CONTROL ROUTINE FOR THE ISODATA CLUSTERING PROGRAM
C  ... DIRECTS THE READING OF INPUT OF DATA MATRIX
C  ... DIRECTS THE COMPUTATIONS FOR THE CURRENT ITERATION
C  IPRINT.NE.0 = PRINT INPUT  (DEFAULT=0)
C  NPRINT.GT.10 = PRINT PCM AND LIST OF PTS IN CLUSTS  (DEFAULT=5)
C   SPECIAL OUTPUT FOR PATTERN RECOGNITION EXPERIMENTS GOES ON TAPE20.
C      IT CAN BE COPIED TO OUTPUT IF DESIRED.
C-----------------------------------------------------------------------
C-----COMMON STORAGE FOR ISODATA WITH THESE BOUNDS...                    D 150
C-----MAXIMUM NUMBER OF PATTERNS = 1000
C-----MAXIMUM DIMENSIONS = 4
C-----MAXIMUM NUMBER OF CLUSTERS = 50                                   D 180
      LOGICAL VERBOSE,WEIGHT
      COMMON /LUNS/  LCARD,LPRINT,LPUNCH
      COMMON /STATS/ NOSCAL,SCALE(4),XAVE(4),XSDV(4),XMIN(4),XMAX(4)
      COMMON /DATA/  DATAPT(4,1000),IPCM(1000)
      COMMON /PARAMS/  NROWS ,NCOLS ,NPARTS,NTHETA,THETAC,NCLST ,NRWDSD
     1          ,TSEONE,NPATMX,NDSCRD,TSQERR,ITERAT,SPHRFC,RMSAVD,AVGA
     2          ,AVG   ,NRWSMX,NPRINT,PCTERR,RANERR,VERBOSE,WEIGHT,NEXP
     *          ,NEXTCL
     3          ,CLAVGD(50),RATIO(50),CLOSE (50),DSTMTX(50),NPTSCL(50)
     4          ,RATMIN(50),DSTNDV(50),AVDST(50),LABCLU(50)
     5          ,CENTER(50,4),STNDEV(50,4),SUMPAT(50,4),SUM3(50),SUM4(50)
      DIMENSION IFOR(8),HEADER(16),SUMSQ(4,50),SDVONE(4)
      DIMENSION PERROR(20)
      INTEGER DEVICE
      DATA  MASK5/77777777770000000000B/, STARUK/4704050313000000000B/
C-----------------------------------------------------------------------
C     FORMATS
    2     FORMAT (1H+,45X,*JOB *A7*  DATE *A10,A6*TIME*I3*:*I2*:*I2,5X
     *         ,*START   CP*F7.3*   PP*F7.3)
    3     FORMAT (8A10)
    4     FORMAT (50F1.0)
    5     FORMAT (*0INITIAL CLUSTER CENTERS AS READ FROM CARDS*)
    6     FORMAT (1X,I5,4X,10F10.2,/,(10X,10F10.2))
    7     FORMAT (1X,I3,* INITIAL CC-S GENERATED USING A SPHERE FACTOR *
     *         ,*OF* G15.6)
    8     FORMAT (1X,8A10)
    9     FORMAT (4I5,F10.0)
   10     FORMAT (1H0,15X,*TOTAL SQUARED ERROR*25X*LUMP* /
     *          * ITER CLUSTERS  PER CENT      RANDOM*
     *          3X*THETAN THETAC NCLST  SPLIT  PATTERNS DISGARDS*)
   11     FORMAT (1X,I3,3X,I5,2X,F9.3,F12.3,I7,F9.1,I5,3X,A7,I7,3X,I6)
   12     FORMAT (I1,F7.0,I2,2A10)
   13     FORMAT (A10,2I5,F10.0,A10)
   14     FORMAT (I5,F5.2)
   15     FORMAT (10X,2I5)
   16     FORMAT (2H *,7A10,A8)
   17     FORMAT (*0PRINTOUT OF DATA MATRIX, UNSCALED VALUES*)
   20     FORMAT (*0THE NUMBER OF ITERATIONS HAS REACHED*I3* -- THAT IS*
     *         ,* THE MAXIMUM SET AT THIS TIME*)
   21     FORMAT (1H0,120(1H*),/,* CONVERGED AFTER*I3* ITERATIONS*)
C-----------------------------------------------------------------------
C     INITIALIZE
C          LOGICAL UNIT NUMBERS
           LCARD   =  5
           LPRINT  =  6
           LPUNCH  =  7
```

```
      LDATA   = LCARD
C     MAXIMUM ARRAY SIZES
      NRWSMX  = 50
      MXROWS  = 50
      MXCOLS  =  4
      MXPATS  = 1000
C     MISCELANEOUS
      NPRINT  = 5
      DEVICE  = 1
      VERBOSE = .FALSE.
      WEIGHT  = .FALSE.
      NROWS   = NEXTCL  =  0
      CCCPRT  = 0.0
      CALL MEMSETX(0.0,CLAVGD,MXROWS)
C-----------------------------------------------------------------------
C     READ IN INFORMATION WHICH IS USED FOR ALL ITERATIONS
  100 IPRINT  = ICARD=NCCOLD=MODE  =  0
      MXITER  = 10
  200 READ (LCARD,3)  IFOR
      IF (EOF(LCARD) .NE. 0.0       )  GO TO 1000
      IF (IFOR(1) .EQ.  3HEND       )  GO TO 200
      IF (IFOR(1) .EQ.  5HPRINT     )  GO TO 210
      IF (IFOR(1) .EQ.  7HVERBOSE   )  GO TO 220
      IF (IFOR(1) .EQ.  9HDIST/SDV  )  GO TO 230
      IF (IFOR(1) .EQ.  9HDIST/DEV  )  GO TO 230
      IF (IFOR(1) .EQ.  3HCCC       )  GO TO 240
C     CHECK FOR UPDATE INPUT *DECK DECLARATION
      IF ((IFOR(1).AND.MASK5) .EQ. STARDK)  GO TO 200
      ICARD  =  ICARD + 1
      IF (ICARD .GT. 2)           GO TO 300
C       FIRST TWO CARDS ARE FOR LABELING PRINTOUT
        CALL MEMVEX(IFOR,HEADER(8*ICARD-7),8)
        GO TO 200
C     SPECIFY AMOUNT OF PRINTOUT
  210 DECODE (20,15,IFOR)  IPRINT,NPRINT
      IF (IFOR(2) .EQ. 1H )  IPRINT = NPRINT = 1
      GO TO 200
C     SET FLAG TO GET MUCH PRINTOUT ON EACH ITERATION
  220 VERBOSE  =  .TRUE.              $  GO TO 200
C    USE DISTANCE DIVIDED BY CLUSTER STANDARD DEVIATION AS MEASURE OF
C     CLOSENESS FOR DETERMINING CLUSTER MEMBERSHIPS.
  230 WEIGHT  =  .TRUE.
      DECODE (10,14,IFOR(2))  NEXP,PCTONE
      IF (NEXP   .EQ.   0)  NEXP   = 2
      IF (PCTONE .LE. 0.0)  PCTONE = 0.10
      GO TO 200
C    SET FLAG SO CLUSTER CHARACTERISTIC CURVE IS PRINTED
  240 CCCPRT  =  1.0                  $  GO TO 200
C    CARD WITH INITIAL PARAMATERS
  300 DECODE (30,9,IFOR)  NCOLS,NPARTS,NPATMX,NRWDSD,SPHRFC
      CALL CCCIN(NULL)
      CALL CLOCKX(IHR,IMIN,ISEC)
      CALL DATEX(D1,D2)
      JOBID   =  MYCPX(17)
      CPZERO  =  SECOND(NULL)
      PPZERO  =  PPTIME(NULL)
      PRINT 1   $   PRINT 2,JOBID,D1,D2,IHR,IMIN,ISEC,CPZERO,PPZERO
      PRINT 8, HEADER
      WRITE (20,)
      WRITE (20,16)  HEADER
      IF (NPATMX .LE. 0)  NPATMX = MXPATS
C     RUN TIME FORMAT
```

```
      READ (LCARD,3)  IFOR
C     SCALE FACTORS
      READ (LCARD, 4)  (SCALE(I),I=1,NCOLS)
C     INITIALIZE DATA SET STATISTICS
      NOSCAL  =  1
      DO 310  I=1,NCOLS
        XAVE(I)  =  XSDV(I)  =  0.0
        XMAX(I)  =  -1.0E20
        XMIN(I)  =   1.0E20
        IF (SCALE(I) .EQ. 0.0)  SCALE(I) = 1.0
        IF (SCALE(I) .NE. 1.0)  NOSCAL   = 0
310   CONTINUE
C    READ THE DATA MATRIX
      IF (IPRINT.NE.0)  WRITE (LPRINT,17)
      DO 320  I=1,NPATMX
        READ (LDATA,IFOR) (DATAPT(J,I),J=1,NCOLS)
        IF (EOF(LDATA) .NE. 0.0)  GO TO 330
        IF (DATAPT(1,I)+DATAPT(2,I) .EQ. 0.0)  GO TO 330
        IPCM(I)  =  1
        IF (IPRINT.NE.0)  WRITE (LPRINT,6) I,(DATAPT(J,I),J=1,NCOLS)
        DO 315  J=1,NCOLS
          X  =  DATAPT(J,I)
          XAVE(J)  =  XAVE(J) + X
          XSDV(J)  =  XSDV(J) + X*X
          IF (X .GT. XMAX(J))  XMAX(J) = X
          IF (X .LT. XMIN(J))  XMIN(J) = X
315     DATAPT(J,I)  =  X/SCALE(J)
320   CONTINUE
      I        =  NPATMX + 1
330   NPATMX   =  I - 1
      TSEONE   =  0.0
      DO 340  I=1,NCOLS
        XAVE(I)  =  XAVE(I)/NPATMX
        X        =  XSDV(I) - XAVE(I)**2*NPATMX
        TSEONE   =  TSEONE +  X
        XSDV(I)  =  SQRT(X/(NPATMX-1))
C       SDVONE -- THE SIZE OF A CLUSTER CONTAINING ONE POINT
C                 SET TO PCTONE (PROPORTION) OF THE DIRECTIONAL STAN. DEV.
        SDVONE(I)  =  XSDV(I)*PCTONE
340   CONTINUE
      IF (IPRINT.NE.0)  PRINT 1
      IF (IPRINT.NE.0)  PRINT 8, HEADER
C    PLOT THESE DATA
      IF (XMIN(1).GE.0.0 .AND. XMIN(1).LE.120.0)  GO TO 360
C     SET BOUNDS OF PRINT PLOT FOR PATTERN RECOGNITION EXPERIMENTS DATA
        XMIN(1)  =  XMIN(2)  =  -2047
        XMAX(1)  =  XMAX(2)  =   2047
        GO TO 400
C     SET BOUNDS FOR CLOUD DATA
360     XMIN(1) = XMIN(2) = 0.0  $  XMAX(1)=120.0  $  XMAX(2)=70.0
400   CALL PLTPT(DATAPT,NPATMX)
C-----------------------------------------------------------------------
C    LOOP FOR EACH ITERATION BEGINS HERE
      ITERAT  =  1
      NDSCRD  =  0
C     READ IN DATA THAT CAN BE CHANGED FOR EACH NEW ITERATION
500   READ (LCARD,3)  IFOR
      IF (EOF(LCARD) .EQ. 0.0)  GO TO 510
C       END-OF-FILE AT ITERATION 1 IS MEANINGLESS
        IF (ITERAT .LE. 1)  GO TO 500
        GO TO (100,910), MODE+1
510   IF (IFOR(1) .EQ. 10H **********)  GO TO 500
```

```
          IF (IFOR(1) .EQ.  5HUSECC      )  GO TO 550
          IF (IFOR(1) .EQ.  6HINITCC     )  GO TO 520
          IF (IFOR(1) .EQ.  5HPUNCH      )  GO TO 530
          IF (IFOR(1) .EQ.  5HSPLIT      )  GO TO 540
          IF (IFOR(1) .EQ.  4HLUMP       )  GO TO 540
          IF (IFOR(1) .EQ.  3HEND        )  GO TO (100,910),MODE+1
          GO TO 600
C
C         ------------------------------------------------
C         READ INITIAL CLUSTER CENTERS
  520       DECODE (20,15,IFOR)  NROWS
            READ (LCARD,3)  IFOR
            WRITE (LPRINT,5)
            DO 522  I=1,NROWS
              READ (LCARD,IFOR)  (CENTER(I,J),J=1,NCOLS)
              WRITE (LPRINT,6)  I,(CENTER(I,J),J=1,NCOLS)
              LABCLU(I)  =  I
              DO 522  J=1,NCOLS
                CENTER(I,J)  =  CENTER(I,J) / SCALE(J)
  522       CONTINUE
          NTHETA  =  LSPARM  =  NCLST  =  THETAC  =  0.0
          NCCOLD  =  NEXTCL  =  NROWS  =  MIN0(MXROWS,NROWS)
            GO TO 700
C         PUNCH CC-S
  530       CALL OUTPUT(2,HEADER,SDVONE)          $  GO TO 500
C         SPLIT OR LUMP UNTIL TWO CONSECUTIVE ITERATIONS HAVE THE
C         SAME NUMBER OF CLUSTERS
  540       DECODE (40,13,IFOR)  LSPARM,NCLST,NTHETA,THETAC,LCCPUN
            MODE  =  1
            GO TO 610
C         USE CCS FROM FIRST DATA SET AS INITIAL ONES ON SECOND -- PUNCH
  550       GO TO 700
C         ------------------------------------------------
C
C     INTERPRET INFORMATION FOR THIS ITERATION
  600     DECODE (30,12,IFOR)  NCLST,THETAC,NTHETA,LSPARM,LCCPUN
  610     NCCOLD  =  NROWS
          IF (.NOT.VERBOSE)  GO TO 615
            PRINT 1
            PRINT 8, HEADER
  615     IF (ITERAT .GT. 1)  GO TO 620
C     COMPUTE INITIAL CLUSTER CENTERS USING SPHERE FACTOR
          CALL INITCC
          NCCOLD  =  0
          WRITE (LPRINT,7)  NROWS,SPHRFC
          GO TO 700
C     DELETE SMALL CLUSTER CENTERS ON ALL BUT THE FIRST ITERATION.
  620     CALL REMOVE(NTHETA)
C     DECIDE WHETHER TO SPLIT OR LUMP (OR NEITHER) . . . . . .
C         USER CAN DECIDE TO SPLIT, LUMP OR SETTLE OR LET PROGRAM DECIDE...
C         IF NROWS ≥ 2*NRWDSD THEN LUMP.  IF NROWS ≤ NRWDSD/2 THEN SPLIT
C         IF NEITHER CONDITION HOLDS, THEN LUMP ON EVEN NUMBERED ITERATIONS
C         AND SPLIT ON ODD ITERATIONS
          IF (LSPARM .EQ. 4HLUMP  )  GO TO 640
          IF (LSPARM .EQ. 5HSPLIT )  GO TO 630
          IF (LSPARM .EQ. 6HSETTLE)  GO TO 700
          IF (NROWS  .GE. 2*NRWDSD)  GO TO 640
          IF (NROWS  .LE. NRWDSD/2)  GO TO 630
          IF (AND(ITERAT,1) .NE. 0)  GO TO 640
C     SPLIT CLUSTERS WHICH SATISFY SPLITTING CRITERIA.
  630     CALL SPLIT(SUMSQ)
          GO TO 700
```

```
C     LUMP PAIRS OF CLUSTERS WHICH SATISFY LUMPING CRITERIA. AT
C     MOST NCLST PAIRS OF CLUSTERS WILL BE LUMPED IN THIS ITERATION.
  640     CALL LUMP(SDVONE)
C    SETTLE
  700     CONTINUE
          CALL MEMSETX(99.99,PERROR,20)
          DO 800  IPART=1,NPARTS
C           PARTITION THE DATA SET
            CALL CLUST(SDVONE,SUMSQ)
C           COMPUTE NEW CLUSTER CENTERS
            DO 730  J=1,NROWS
              IF (NPTSCL(J) .EQ. 0)  GO TO 730
              D  =  1.0 / NPTSCL(J)
              DO 720  K=1,NCOLS
                IF (.NOT.WEIGHT)  GO TO 720
                STNDEV(J,K)  =  SQRT((SUMSQ(K,J)-D*SUMPAT(J,K)**2)*D)
  720           CENTER(J,K)  =  SUMPAT(J,K)*D
  730       CONTINUE
C           REMOVE EMPTY CLUSTERS
            CALL REMOVE(0)
            IF (.NOT.VERBOSE)  GO TO 790
            PERROR(IPART)  =  TSQERR/TSEONE*100.0
            PERROR(IPART+10)  =  PCTERR
            PRINT 740 , ITERAT,IPART,NROWS,PERROR(IPART),PCTERR
  740       FORMAT (*0ITERATION,PARTITION*2I5 *   NROWS*I3
     +           ,*     ERRORS*2G20.6)
            DO 750  I=1,NROWS
              CALL WRCC(I,7,1)
  750       CONTINUE
  751       FORMAT (1X,40I3)
            PRINT 751, (IPCM(I),I=1,NPATMX)
  790       IF (NROWS .EQ. 1)  GO TO 810
  800     CONTINUE
C    COMPUTE STATISTICS
  810     CALL STATS
          NN  =  NPATMX - NDSCRD
          IF (VERBOSE)                                  PRINT 10
          IF (.NOT.VERBOSE .AND. ITERAT.EQ.1)  PRINT 10
          PRINT 11, ITERAT,NROWS ,PCTERR,RANERR
     +             ,NTHETA,THETAC,NCLST ,LSPARM,NN    ,NDSCRD
          IF (.NOT.VERBOSE)  GO TO 820
C    PRINT STATISTICS
          DEVICE=1    $   IF (LCCPUN.EQ.5HPUNCH) DEVICE=3
          CALL OUTPUT(DEVICE,HEADER,SDVONE)
          PRINT 821, (IPART,IPART=1,NPARTS)
          PRINT 822, (PERROR(IPART),IPART=1,NPARTS)
  821     FORMAT (*0ERROR FOR EACH SETTLING PARTITION*/(1X,I8,9I12))
  822     FORMAT (1X,F8.2,9F12.2)
          PRINT 823, (PERROR(I+10),I=1,NPARTS)
  823     FORMAT (1X,10G12.4)
          CALL PLTCC(CENTER,NROWS)
  820     ITERAT  =  ITERAT + 1
          IF (MODE .GT. 0)  GO TO 900
          IF (ITERAT.GT.3 .AND. NROWS.EQ.1)  STOP1111
          IF (ITERAT .LE. MXITER)  500,910
C     SPLIT (LUMP) UNTIL THE NUMBER OF CLUSTERS CONVERGES
  900     IF (NROWS .EQ. NCCOLD)  500,610
C    CHECK IF OPTION TO PRINT (PUNCH) ON LAST ITERATION IS USED
  910     IF (.NOT.VERBOSE)  CALL OUTPUT(1,HEADER,SDVONE)
          IF (.NOT.VERBOSE)  CALL PLTCC(CENTER,NROWS)
          ITERAT  =  ITERAT - 1
          IF (MODE .GT. 0)  PRINT 21, ITERAT
```

```
      IF (MODE .EQ. 0)  PRINT 20, ITERAT
      IF (CCCPRT .NE. 0.0)  CALL CCCPR(DATAPT)
      GO TO 100
C-----------------------------------------------------------------------
 1000 CONTINUE
      PRINT 1
      END
```

```
      SUBROUTINE CLUST(SDVONE,SUMSQ)
      COMMON /OCT26/  XXXXXX
C-----PARTITION THE SET OF PATTERNS                                          G 110
C-----
      LOGICAL VERBOSE,WEIGHT,NOTWGT
      COMMON /DATA/  DATAPT(4,1000),IPCM(1000)
      COMMON /PARAMS/  NROWS ,NCOLS ,NPARTS,NTHETA,THETAC,NCLST ,NRWDSD
     1          ,TSEONE,NPATMX,NDSCRD,TSQERR,ITERAT,SPHRFC,RMSAVD,AVGA
     2          ,AVG   ,NRWSMX,NPRINT,PCTERR,RANERR,VERBOSE,WEIGHT,NEXP
     *          ,NEXTCL
     3          ,CLAVGD(50),RATIO(50),CLOSE (50),DSTMTX(50),NPTSCL(50)
     4          ,RATMIN(50),OSTNDV(50),AVDST(50),LABCLU(50)
     5          ,CENTER(50,4),STNDEV(50,4),SUMPAT(50,4),SUM3(50),SUM4(50)
      REAL MINDST,DATA(4),SUMSQ(4,50),SDVONE(4)
C     INITIALIZE
          TSQERR  =  PCTERR  =  0.0
          CALL MEMSETX(0,NPTSCL,NROWS)
          IF (WEIGHT)  CALL MEMSETX(0.0,SUMSQ,4*NROWS)
          CALL MEMSETX(0.0,SUMPAT,50*4)
          NOTWGT  =  .NOT.WEIGHT
C-------------------------------------------------------------------------
C     ASSIGN EACH POINT TO A CLUSTER
          DO 10  K=1,NPATMX
            IF (IPCM(K) .EQ.0) GO TO 10
            CALL MEMVEX(DATAPT(1,K),DATA,NCOLS)
            MINDST = 1.0 E + 300
            ICLNO  =  0
C           FIND THE CLUSTER CENTER CLOSEST TO PATTERN
            DO 5 I = 1, NROWS
              DSTNC  =  D  =  0.0
              DO 4  J=1,NCOLS
                X    =  (CENTER(I,J) - DATA(J))**2
                D    =  D + X
                IF (NOTWGT)  GO TO  4
                S    =  AMAX1(STNDEV(I,J),SDVONE(J))
                X    =  X / S**NEXP
    4         DSTNC  =  DSTNC + X
              IF (DSTNC .GE. MINDST)  GO TO 5
              ICLNO   =  I
              DMIN    =  D
              MINDST = DSTNC
    5       CONTINUE
C           STORE THE CLUSTER NUMBER FOR THIS PATTERN
            IPCM(K) = ICLNO
            TSQERR  =  TSQERR + DMIN
            PCTERR  =  PCTERR + MINDST
C           ACCUMULATE SUMS FOR FINDING CLUSTER MEANS
            DO 6 J = 1, NCOLS
              SUMPAT(ICLNO,J) = SUMPAT(ICLNO,J) + DATA(J)
              IF (WEIGHT)  SUMSQ(J,ICLNO) = SUMSQ(J,ICLNO) + DATA(J)**2
    6       CONTINUE
C           COUNT THE PATTERNS IN THE CLUSTER
            NPTSCL(ICLNO) = NPTSCL(ICLNO) + 1
   10     CONTINUE
C-------------------------------------------------------------------------
C     TO HERE WHEN ALL PATTERNS HAVE BEEN READ
          RETURN
      END
```

```
      SUBROUTINE LUMP(SDVONE)
      COMMON /OCT26/  XXXXXX
C     COMBINE CLUSTER CENTERS WHICH ARE CLOSER THAN THETAC UNITS APART.
C     COMBINE AT MOST MIN(19,NCLST) PAIRS OF CLUSTER CENTERS IN ONE ITERATION
C     IF WEIGHT IS TRUE, USE MAHALANOBIS DISTANCE IN PLACE ON EUCLIDIAN.
C     SDVONE IS THE STANDARD DEVIATION OF A CLUSTER CONTAINING ONE POINT
C-----------------------------------------------------------------------
      LOGICAL VERBOSE,WEIGHT
      COMMON /PARAMS/  NROWS ,NCOLS ,NPARTS,NTHETA,THETAC,NCLST ,NRWDSD
     1          ,TSEONE,NPATMX,NDSCRD,TSQERR,ITERAT,SPHRFC,RMSAVD,AVGA
     2          ,AVG   ,NRWSMX,NPRINT,PCTERR,RANERR,VERBOSE,WEIGHT,NEXP
     *          ,NEXTCL
     3          ,CLAVGD(50),RATIO(50),CLOSE (50),DSTMTX(50),NPTSCL(50)
     4          ,RATMIN(50),USTNDV(50),AVDST(50),LABCLU(50)
     5          ,CENTER(50,4),STNDEV(50,4),SUMPAT(50,4),SUM3(50),SUM4(50)
C
      COMMON             PUL(20),IPUL(2,20),IUSED1(50),CEN(4),DEV(4)
     *                   ,DUMMY(232)
      DIMENSION SDVONE(4)
C-----------------------------------------------------------------------
C     ARRANGE THE CLUSTERS IN PAIRS, ORDERED FROM CLOSEST THGETHER.  PUL IS
C     THE DISTANCE BETWEEN THE CLUSTERS GIVEN BY IPUL(1) AND IPUL(2).
C     IF WEIGHT IS TRUE, USE MAHALANOBIS DISTANCE, OTHERWISE EUCLIDIAN.
C          PRINT 101, NROWS,NTHETA,NCLST,NCOLS,THETAC
C 101     FORMAT (*1LUMP*4I5,F10.2)
          NTBL  =  0
          CALL MEMSETX( 0 ,IPUL,40)
          CALL MEMSETX(-5.0,PUL,20)
          TSQ   =  THETAC*THETAC
          NM1   =  NROWS - 1
          DO 190  II=1,NM1
            IP  =  II + 1
            DO 180  JJ=IP,NROWS
C             CALCULATE DISTANCE BETWEEN CLUSTERS II AND JJ
              IF (WEIGHT)  GO TO 120
C             EUCLIDIAN DISTANCE
              DIST  =  0.0
              DO 110  I=1,NCOLS
                K   =  (I-1)*NRWSMX
  110         DIST  =  DIST + (CENTER(K+II) - CENTER(K+JJ))**2
              IF (DIST - TSQ)  180,180,130
C             MAHALANOBIS DISTANCE -- BILATERAL -- KEEP MAXIMUM OF TWO
  120         DIST  =  DIST2  =  0.0
              DO 125  I=1,NCOLS
                K       =  (I-1)*NRWSMX
                DV1     =  AMAX1(STNDEV(K+II),SDVONE(I))
                DV2     =  AMAX1(STNDEV(K+JJ),SDVONE(I))
                X       =  (CENTER(K+II) - CENTER(K+JJ))**2
                DIST    =  DIST  + X / (DV1**NEXP)
                DIST2   =  DIST2 + X / (DV2**NEXP)
  125         CONTINUE
              DIST  =  AMAX1(DIST,DIST2)
C             STORE THE 20 SMALLEST VALUES OF DIST
  130         DO 135  L=1,20
                IF (PUL(L) .LT. 0.0 )  GO TO 150
                IF (PUL(L) .GT. DIST)  GO TO 140
  135         CONTINUE
C             TABLE USED UP WITH SMALLER VALUES
              GO TO 180
C             MOVE ALL PAIRS IN IPUL DOWN
  140         IF (L .GT. 19)  GO TO 150
              DO 145  I=L,19
```

```
                J  =  19 + L - I
                IPUL(1,J+1)  =  IPUL(1,J)
                IPUL(2,J+1)  =  IPUL(2,J)
                PUL (  J+1)  =   PUL(  J)
  145         CONTINUE
  150         NTBL         =  MIN0(20,NTBL+1)
              IPUL(1,L)    =  II
              IPUL(2,L)    =  JJ
              PUL (  L)    =  DIST
  180       CONTINUE
  190     CONTINUE
C         PRINT 191, NTBL,((IPUL(I,J),J=1,20),I=1,2)
C 191     FORMAT (10X,*NTBL*,I5/* IPUL*,20I6/5X,20I6/*  PUL*,20F6.0)
C-----------------------------------------------------------------------
C   USING THE LIST OF POSSIBLE PAIRS OF CLUSTERS TO LUMP, IPUL, LUMP
          CALL MEMSETX(0,IUSED1,NROWS)
          NLUMP  =  0
          IF (WEIGHT)  GO TO 230
C   LUMP AT MOST NCLST PAIRS WHICH ARE CLOSER THAN THETAC UNITS APART
C   IN EUCLIDIAN DISTANCE.
          DO 220  L=1,NTBL
            I    =  IPUL(1,L)      $    IF (I .EQ. 0)  GO TO 300
            J    =  IPUL(2,L)      $    IF (J .EQ. 0)  GO TO 300
            IF (IUSED1(I).EQ.1 .OR. IUSED1(J).EQ.1)  GO TO 220
            P1   =  NPTSCL(I)
            P2   =  NPTSCL(J)
            NPTSCL(I)  =  P  =  P1 + P2
            NPTSCL(J)  =  0
            DO 210  K=1,NCOLS
              IK  =  (K-1)*NRWSMX + I
              JK  =  (K-1)*NRWSMX + J
              CENTER(IK)  =  (CENTER(IK)*P1 + CENTER(JK)*P2)/P
              STNDEV(IK)  =  SQRT((STNDEV(IK)**2*(P1-1.)
     +                     + STNDEV(JK)**2*(P2-1.))/(P-1.0))
  210       CONTINUE
            NLUMP       =  NLUMP + 1
            CLAVGD(I)   =  (CLAVGD(I)*P1 + CLAVGD(J)*P2) / P
            IUSED1(I)   =  IUSED1(J)  =  1
            IF (.NOT.VERBOSE)  GO TO 215
            IF (NLUMP .EQ. 1)  PRINT 1001, ITERAT,THETAC,NCLST
            PRINT 1002, I,J
  215       IF (NLUMP .GE. NCLST)  GO TO 300
  220     CONTINUE
          GO TO 300
C   WHEN USING MAHALANOBIS DISTANCE, APPLY KOLMOGOROV-SMIRNOV TEST TO
C   SEE IF CLUSTERS SHOULD BE JOINED.
C   (REF.  GIBBONS, JEAN D., NONPARAMETRIC STATISTICAL INFERENCE,
C          MCGRAW-HILL, PP 75-88)
  230     CONTINUE
          DO 280  L=1,NTBL
            I    =  IPUL(1,L)      $    IF (I .EQ. 0)  GO TO 300
            J    =  IPUL(2,L)      $    IF (J .EQ. 0)  GO TO 300
            IF (IUSED1(I).EQ.1 .OR. IUSED1(J).EQ.1)  GO TO 280
            P1   =  NPTSCL(I)
            P2   =  NPTSCL(J)
            P    =  P1 + P2
C           FORM ESTIMATES OF POOLED CENTER AND DEVIATIONS
            DO 240  K = 1,NCOLS
              IK       =  (K-1)*NRWSMX + I
              JK       =  (K-1)*NRWSMX + J
              CEN(K)   =  (CENTER(IK)*P1 +CENTER(JK)*P2)/P
              DEV(K)   =  SQRT((STNDEV(IK)**2*(P1-1.)
```

```
     *                          * STNDEV(JK)**2*(P2-1.))/(P-1.0))
  240        CONTINUE
C            FOR PRESENT VERSION (JULY 18, 1972), LUMP AT MOST NCLST
C            PAIRS CLOSER THAN THETAC UNITS.  THETAC IS A MAHALANOBIS DIST
             IF (PUL(L) .GE. THETAC*THETAC)  GO TO 300
             NLUMP  =  NLUMP + 1
             DO 250  K=1,NCOLS
               CENTER(I,K)  =  CEN(K)
               STNDEV(I,K)  =  DEV(K)
  250        CONTINUE
             NPTSCL(I)  =  P
             NPTSCL(J)  =  0
             CLAVGD(I)  =  (CLAVGD(I)*P1 + CLAVGD(J)*P2) / P
             IUSED1(I)  =  IUSED1(J)  =  1
             IF (.NOT.VERBOSE)  GO TO 270
             IF (NLUMP .EQ. 1)  PRINT 1001, ITERAT,THETAC,NCLST
             PRINT 1002, I,J
  270        IF (NLUMP .GE. NCLST)  GO TO 300
  280      CONTINUE
C-----------------------------------------------------------------------
C    MOVE CLUSTERS UP IN ARRAYS
  300      CONTINUE
           CALL REMOVE(0)
           RETURN
C-----------------------------------------------------------------------
 1001      FORMAT (14H0IN ITERATION ,I3,14H WITH THETA = ,F10.2
     *          ,13H AND NCLST = ,I2,1H.)
 1002      FORMAT (10X,10H CLUSTERS ,I2,5H AND ,I2
     *          ,22H WERE LUMPED TOGETHER.)
C-----------------------------------------------------------------------
       END
```

Listing for Program PLACE

```
C  PLACE CLUSTER CENTERS IN A SET OF GENERATED DATA
C          JULY 21, 1972
C  MAXCC   MAXIMUM NUMBER OF CLUSTERS
C  LP      LOGICAL UNIT NUMBER FOR LINE PRINTER
C  KB      LOGICAL UNIT NUMBER FOR TELETYPE
C  KR      LOGICAL UNIT NUMBER FOR INPUT FILE
C  NEWCCS FLAG .NE. 0 IF NEW CLUST SPECS ENTERED SINCE GENERATING PICT
C  CLUSTER SPECIFICATIONS -- 3 FIELDS IN NAME.....
C    X,Y     DENOTE HORIZONTAL OR VERTICAL COMPONENT, RESP.
C    CC,DV   CLUSTER CENTER OR STANDARD DEVIATION
C    O,G,S   SPECIFIED BY OPERATOR (ORDAINED), GENERATED (CALCULATED)
C            OR S FOR SUBJECT'S RESPONSE
C    NPT     NUMBER OF POINTS IN CLUSTER
C  LPRINT BUFFER FOR PRINT PLOT OF SCREEN IMAGE
C  NCCS    NUMBER OF CLUSTER CENTERS DETECTED BY SUBJECT
C-----------------------------------------------------------------------
      COMMON /DEBUG/  LDEBUG                                              1
      COMMON /RANDS/  IRAND1,IRAND2                                       2
      COMMON /LUNS/  KB,LP,KR                                             3
      COMMON /TXTBUF/  LUNTXT,MAXTXT,ICR,ITXTBF(80)                       4
      COMMON /INDPLT/  INDPLT(51)                                         5
      COMMON  /SUBJ/  NCCS,XCCS(50),YCCS(50)                              6
      COMMON /BOUNDS/  MINH,MAXH,MINV,MAXV,HLEN,VLEN,IDOT                7
      COMMON  /STATS/  NCCO                                               8
     1          ,NPTO(50),XCCO(50),YCCO(50),XDVO(50),YDVO(50)
     2          ,NPTG(50),XCCG(50),YCCG(50),XDVG(50),YDVG(50),COR(50)
      DIMENSION IDATE(3),PRTPLT(1200),IPLACE(16),MOUSEP(7)                9
      DATA IPLACE/2H P,2HLA,2HCE,2H B,2HUG,2HMA,2HRK,2H A,2HT ,2HCL      10
     .          ,2HUS,2HTE,2HR ,2HCE,2HNT,2HER  /
      DATA MOUSEP/2H  ,2H  ,2H  ,2H  ,2H  ,2H  ,2H  /                    11
C-----------------------------------------------------------------------
    1 FORMAT (1H1)                                                       12
    2 FORMAT (6H DATE ,A2,1H-,A2,1H-,A2)                                 13
    3 FORMAT (' COMMANDS:  C,D,E,G,H,I,L,N,O,P,R,Z',/)                   14
    4 FORMAT (38H1LIST OF COMMANDS FOR PLACE, 16 DEC 71     ,/           15
     1   ,5X,33HC -- CLUSTER CENTER FOR CLUSTER I             ,/
     2   ,5X,37HD -- STANDARD DEVIATION FOR CLUSTER I         ,/
     3   ,5X,20HE -- EXIT TO MONITOR                           )
    5 FORMAT (5X,24HG -- GO, START EXECUTION                  ,/        16
     1   ,5X,4HI --                                            )
    6 FORMAT (5X, 4HL --                                      ,/        17
     1   ,5X, 4HN --                                          ,/
     2   ,5X, 4HO --                                          ,/
     3   ,5X, 4HP --                                          ,/
     4   ,5X, 4HR --                                           )
    7 FORMAT (5X, 4HZ --                                       /)       18
   11 FORMAT (80A1)                                                      19
   12 FORMAT (' CLUSTER',I3,' OF',I3,5X,'NPT',I4                         20
     .          ,5X,' CC ',2F6.0,5X,'DEV',2F5.0,/)
   14 FORMAT (1H0,80A1)                                                  21
   20 FORMAT (55H0SUMMARY OF DIFFERENCES BETWEEN SUBJECT'S CENTERS AND G 22
     1    ,'ENERATED CENTERS',/
     2    ,28X,'A V E R A G E S          STANDARD DEVIATION',/
     3    ,9X,'REPLICATIONS',7X,  2(1HX,6X,1HY,5X,4HDIST,8X))
   21 FORMAT (10X,I5,8X,2F7.0,F8.0,3X,2F7.0,F8.0)                        23
   22      FORMAT (' FINAL RANDOM NUMBERS',2I7)                          24
   30 FORMAT ('0THEORETICAL',/                                           25
     1    ,5X,'NUMBER CLUSTERS',I3,7X,'BEGINNING RANDOM NUMBER',2I7
     2    ,/,13X,'O P E R A T O R   S P E C S',9X
     2          ,'A S    G E N E R A T E D',/
     3    ,7X,2(14X,'CENTER       STAN DEV'),/
     4    ,5X,'CLUST NPTS',2(6X,1HX,6X,1HY),4X,4HNPTS,2(6X,1HX,6X,1HY) )
```

```
   31 FORMAT (5X,I4,I6,1X,2F7.0,   2F7.0,3X,I4,1X,2F7.0,   2F7.0)              26
C  31 FORMAT (5X,I4,I6,1X,2F7.0,1X,2F6.0,3X,I4,1X,2F7.0,1X,2F6.0)
   40 FORMAT(#0TRIAL#,I3,/,#0SUBJECT RESPONSE#,/,# NUMBER CLUSTERS#,I3,/      27
     1    ,15X,#CENTER#10X,#CLOSEST  TO#,8X,#DIFFERENCE#,/
     2    ,5X,#CLUST    X      Y#,5X,#NO     X      Y#,7X
     2         ,#X     Y    DIST#)
   41 FORMAT (5X,I4,1X,2F7.0, I6,2F7.0,2X,2F6.0,F7.0)                         28
   61 FORMAT (# NOT ENOUGH VALUES IN INPUT TEXT#,/)                           29
C-------------------------------------------------------------------
C    INITIALIZE
          IBELL   = 03407                                                   30
          LBNDRY  = 0                                                       31
          CALL INIT(INDBOX,IDATE)                                           32
          WRITE (LP,2) IDATE                                                33
          WRITE (KB,3)                                                      34
          ICR  = 1H$                                                        35
C    SET DEFAULT VALUES
  800     NCCO   = 1                                                        36
          ICC    = 1                                                        37
          NEWCCS = 1                                                        38
          NCCG   = 0                                                        39
          MAXCC  = 50                                                       40
          DO 810  I=1,MAXCC                                                 41
            YCCO(I)  = 0.0                                                  42
            XCCO(I)  = 0.0                                                  43
            XDVO(I)  = 300.                                                 44
            YDVO(I)  = 300.                                                 45
            NPTO(I)  = 100                                                  46
  810     CONTINUE                                                          47
C    ZERO SUMMARY STATISTICS
  900     SUMDX  = 0.0                                                      48
          SUMDY  = 0.0                                                      49
          SSQDX  = 0.0                                                      50
          SSQDY  = 0.0                                                      51
          SUMDD  = 0.0                                                      52
          SSQDD  = 0.0                                                      53
          NTIME  = 0                                                        54
          LASTJJ = 0                                                        55
C-------------------------------------------------------------------
C    OPERATOR CAN DEFINE CLUSTERS.  N IS A COUNTER OF CHARACTERS IN
C         INPUT TEXT BUFFER ITXTBF
 1000     CONTINUE                                                          56
          CALL RDTXT(N)                                                     57
 1005     IF (LASTJJ .LE. 0)  GO TO 1006                                    58
            JJ  = LASTJJ                                                    59
            GO TO 1007                                                      60
 1006     N  = N + 1                                                        61
          IF (N .GT. MAXTXT)  GO TO 1000                                    62
          JJ  = ITXTBF(N)                                                   63
 1007     LASTJJ  = 0                                                       64
          IF (JJ .EQ. 1H )  GO TO 1005                                      65
          IF (JJ .EQ. 1HB)  GO TO 1020                                  *   66
          IF (JJ .EQ. 1HC)  GO TO 1030                                      67
          IF (JJ .EQ. 1HD)  GO TO 1040                                      68
          IF (JJ .EQ. 1HE)  GO TO 9000                                      69
          IF (JJ .EQ. 1HG)  GO TO 2000                                      70
          IF (JJ .EQ. 1HH)  GO TO 1080                                  *   71
          IF (JJ .EQ. 1HI)  GO TO 1090                                      72
          IF (JJ .EQ. 1HL)  GO TO 1120                                  *   73
          IF (JJ .EQ. 1HN)  GO TO 1140                                      74
          IF (JJ .EQ. 1HO)  GO TO 1150                                      75
          IF (JJ .EQ. 1HP)  GO TO 1160                                      76
```

```
            IF (JJ .EQ. 1HQ)  GO TO 1170                                        77
            IF (JJ .EQ. 1HR)  GO TO 1180                                        78
            IF (JJ .EQ. 1HS)  GO TO 1190                                    *   79
            IF (JJ .EQ. 1HW)  GO TO 1230                                    *   80
            IF (JJ .EQ. 1HZ)  GO TO 1260                                        81
C           THE FOLLOWING CHARACTERS ASSUME THE REST OF THE LINE IS A COMMENT
            IF (JJ .EQ. 1H.)  GO TO 1000                                    *   82
            IF (JJ .EQ. 1H,)  GO TO 1000                                    *   83
            IF (JJ .EQ. 1H$)  GO TO 1000                                    *   84
            IF (JJ .EQ. 1H/)  GO TO 1000                                    *   85
C           CHECK FOR END OF TEXT
            IF (JJ .EQ. 1H*)  GO TO 1000                                    *   86
            IF (JJ .EQ. ICR)  GO TO 1000                                        87
C     ILLEGAL CHARACTER
 1008       NP1  =  N + 1                                                       88
            ITXTBF(NP1)  =  1HE                                             **** 89
            WRITE (KB,14)  (ITXTBF(I),I=1,NP1)                                 90
            WRITE (KB,3)                                                        91
            GO TO 1000                                                          92
C     NOT ENOUGH VALUES IN INPUT TEXT
 1009       N  =  INDTXT(ITXTBF,MAXTXT)                                         93
            WRITE (KB,14)  (ITXTBF(I),I=1,N)                                    94
            WRITE (KB,61)                                                       95
            GO TO 1000                                                          96
C     MODIFY SWITCH WHICH ALLOWS SUBJECT TO SPECIFY BOUNDARY
 1020       LBNDRY  =  1 - LBNDRY                                               97
            GO TO 1005                                                          98
C     CLUSTER CENTER
 1030       XCCO(ICC)  =  GETVAL(ITXTBF,N,XCCO(ICC),ICR,IERR)                   99
            NEWCCS  =  1                                                       100
            IF (IERR.GT.0)  GO TO (1009,1008,1310),IERR                        101
            YCCO(ICC)  =  GETVAL(ITXTBF,N,YCCO(ICC),ICR,IERR)                  102
            IF (IERR .EQ. 0)  GO TO 1005                                       103
            GO TO (1000,1008,1009),IERR                                        104
C     CLUSTER STANDARD DEVIATIONS
 1040       XDVO(ICC)  =  GETVAL(ITXTBF,N,XDVO(ICC),ICR,IERR)                  105
            NEWCCS  =  1                                                       106
            IF (IERR.GT.0)  GO TO (1009,1008,1310),IERR                        107
            YDVO(ICC)  =  GETVAL(ITXTBF,N,YDVO(ICC),ICR,IERR)                  108
            IF (IERR .EQ. 0)  GO TO 1005                                       109
            GO TO (1000,1008,1009),IERR                                        110
C     USE OLD CLUST SPECS IF THEY HAVEN'T BEEN CHANGED SINCE MAKING PICT
 1080       IF (NEWCCS .NE. 0)  GO TO 2000                                     111
            IF (NCCG   .EQ. 0)  GO TO 2000                                     112
            GO TO 3000                                                         113
C     CLUSTER INDEX
 1090       OLD  =  ICC                                                        114
            ICC  =  GETVAL(ITXTBF,N,OLD,ICR,IERR)                              115
            IF (IERR .GT. 1)  GO TO (1000,1008,1310),IERR                      116
            IF (ICC.GE.1.AND.ICC.LE.NCCO)  IF (IERR)  1000,1005,1000           117
            IC   =  OLD                                                        118
            WRITE (KB,1091)  ICC,NCCO,IC                                       119
            ICC  =  IC                                                         120
 1091       FORMAT (' TOO LARGE VALUE',I6,5X,'MAX IS',I3,5X,'SET TO',I3)       121
            IF (IERR)  1000,1005,1000                                          122
C     LIST PARAMETERS OF CLUSTER ICC
 1120       WRITE (KB,12)  ICC,NCCO,NPTO(ICC)                                  123
     +                    ,XCCO(ICC),YCCO(ICC),XDVO(ICC),YDVO(ICC)
            GO TO 1005                                                         124
C     NUMBER OF CLUSTERS
 1140       OLD     =  NCCO                                                    125
            NCCO    =  GETVAL(ITXTBF,N,OLD,ICR,IERR)                           126
```

```
            NEWCCS  =  1                                                        127
            IF (IERR .GT. 1)  GO TO (1000,1008,1310),IERR                       128
            IF (NCCO.GE.1.AND.NCCO.LE.MAXCC)  IF (IERR) 1000,1005,1000          129
            IC      =  OLD                                                      130
            WRITE (KB,1091)  NCCO,MAXCC,IC                                      131
            NCCO    =  IC                                                       132
            IF (IERR)  1000,1005,1000                                           133
C     LIST OPTIONS ON LINE PRINTER
 1150       WRITE (LP,4)                                                        134
            WRITE (LP,5)                                                        135
            WRITE (LP,6)                                                        136
            WRITE (LP,7)                                                        137
            GO TO 1005                                                          138
C     NUMBER OF POINTS IN CLUSTER ICC
 1160       OLD         =  NPTO(ICC)                                            139
            NEWCCS      =  1                                                    140
            NPTO(ICC)   =  GETVAL(ITXTBF,N,OLD,ICR,IERR)                        141
            IERR        =  IERR + 1                                             142
            GO TO (1005,1000,1008,1310), IERR                                   143
C     DEBUGGING PRINTOUT SWITCH
 1170       OLD     =  LDEBUG                                                   144
            LDEBUG  =  GETVAL(ITXTBF,N,OLD,ICR,IERR)                            145
            IERR    =  IERR + 1                                                 146
            GO TO (1005,1000,1008,1310), IERR                                   147
C     INITIAL RANDOM NUMBER
 1180       OLD     =  IRAND1                                                   148
            IRAND1  =  GETVAL(ITXTBF,N,OLD,ICR,IERR)                            149
            IF (IERR.GT.0)  GO TO (1009,1008,1310), IERR                        150
            OLD     =  IRAND2                                                   151
            IRAND2  =  GETVAL(ITXTBF,N,OLD,ICR,IERR)                            152
            IERR    =  IERR + 1                                                 153
            GO TO (1005,1000,1008,1009),IERR                                    154
C     SWITCH TEXT INPUT UNITS
 1190       IF (LUNTXT .EQ. KB)  GO TO 1191                                     155
              LUNTXT  =  KB                                                     156
              GO TO 1000                                                        157
 1191         LUNTXT  =  KR                                                     158
              GO TO 1000                                                        159
C     REWIND DISK FILE
 1230       REWIND KR                                                           160
            GO TO 1005                                                          161
C     ZERO CUMULATIVE STATISTICS -- AFTER PRINTING
 1260       IF (NTIME .LE. 1)  GO TO 900                                        162
            R  =  1.0/NTIME                                                     163
            S  =  1.0/(NTIME-1)                                                 164
            SSQDX  =  SQRT((SSQDX - SUMDX*SUMDX*R)*S)                           165
            SSQDY  =  SQRT((SSQDY - SUMDY*SUMDY*R)*S)                           166
            SSQDD  =  SQRT((SSQDD - SUMDD*SUMDD*R)*S)                           167
            SUMDX  =  SUMDX * R                                                 168
            SUMDY  =  SUMDY * R                                                 169
            SUMDD  =  SUMDD * R                                                 170
            WRITE (LP,20)                                                       171
            WRITE (LP,21)  NTIME,SUMDX,SUMDY,SUMDD,SSQDX,SSQDY,SSQDD           172
            WRITE (LP,1)                                                        173
            WRITE (LP,2)  IDATE                                                 174
            GO TO 900                                                           175
C     RETAIN PREVIOUS CHARACTER FROM INPUT STREAM
 1310       LASTJJ  =  JJ                                                       176
            GO TO 1000                                                          177
C-------------------------------------------------------------------------
C     GENERATE CLUSTER(S)
 2000       CONTINUE                                                            178
```

```
      IF (IRAND1.EQ.0 .AND. IRAND2.EQ.0)  CALL NORMAL(START)                179
      CALL SETIND(INDBOX)                                                   180
      CALL PPLOT(PRTPLT,NULL,NULL,1)                                        181
      WRITE (LP,30)  NCCO,IRAND1,IRAND2                                     182
      DO 2100  I=1,NCCO                                                     183
        CALL GENCC(I,PRTPLT)                                                184
        WRITE (LP,31)  I,NPTO(I),XCCO(I),YCCO(I),XDVO(I),YDVO(I)            185
     *                 ,NPTG(I),XCCG(I),YCCG(I),XDVG(I),YDVG(I)
 2100 CONTINUE                                                              186
      CALL GETIND(INDCCS)                                                   187
      NEWCCS  =  0                                                          188
      NCCG    =  NCCO                                                       189
C-----------------------------------------------------------------------
C     PLACE BUGMARKS AT CLUSTER CENTERS
 3000 CONTINUE                                                              190
      WRITE (KB,11)  IBELL                                                  191
      CALL MOVBM(-1200,-2000)                                               192
      CALL CHARS(IPLACE,16,2)                                               193
      CALL GETIND(INDPLT(1))                                                194
      CALL TRKCHR(1H*,1)                                                  * 195
C     CLEAR DISPLAY KEYBOARD BUFFER
      DO 3010  I=1,80                                                       196
 3010 NULL    =  LCHAR(ICH)                                                197
      NCCS  =  0                                                            198
C     THIS LOOP TO TRACK MOUSE
 3100   CALL MOUSE(IH,IV)                                                   199
        IF (LCHAR(ICH) .NE. 0)  GO TO 3300                                  200
        IF (IBUTTN(IVAL).EQ.0)  GO TO 3100                                  201
        IF (IVAL .EQ. 2)  GO TO 3320                                        202
C       TALLY THIS POINT FOR THE SUBJECT
        NCCS  =  NCCS + 1                                                   203
        XCCS(NCCS)  =  IH                                                   204
        YCCS(NCCS)  =  IV                                                   205
C       LEAVE * AT THE POINT
        CALL SETIND(INDPLT(NCCS))                                           206
        CALL MOVBM (IH,IV)                                                  207
        CALL CHARS (1H*,1,1)                                                208
        CALL GETIND(INDPLT(NCCS+1))                                         209
        CALL PPLOT(PRTPLT,IH,IV,1H*)                                      * 210
C       LOOP UNTIL SUBJECT TAKES HIS FINGER FROM THE BUG BUTTON
 3200   IF (IBUTTN(IVAL))  3200,3100,3200                                   211
C    CHECK CHARACTER FROM DISPLAY KEYBOARD
 3300   IF (ICH .EQ. 1H  )  GO TO 4000                                    * 212
        IF (ICH .EQ. 1HA )  GO TO 3310                                    * 213
        IF (ICH .EQ. 1HB )  GO TO 3320                                    * 214
        IF (ICH.EQ.020141)  GO TO 3310                                      215
        IF (ICH.EQ.020142)  GO TO 3320                                      216
        IF (ICH .EQ. 1HS )  GO TO 3330                                    * 217
        GO TO 3100                                                          218
C
C       ABORT THIS TRIAL
 3310   CALL TRKCHR(1H ,3)                                                * 219
        CALL SETIND(INDCCS)                                                 220
        GO TO 1005                                                          221
C       ERASE PREVIOUS SUBJECT CENTER
 3320   IF (NCCS .EQ. 0)  GO TO 3100                                        222
        IH  =  XCCS(NCCS)                                                   223
        IV  =  YCCS(NCCS)                                                   224
        CALL PPLOT(PRTPLT,IH,IV,1H )                                      * 225
        CALL SETIND(INDPLT(NCCS))                                           226
        NCCS  =  NCCS - 1                                                   227
        GO TO 3200                                                          228
```

```
C          SWITCH TEXT INPUT UNITS
 3330        I  =  KB                                                    229
             IF (LUNTXT .EQ. KB)  I = KB                                 230
             LUNTXT  =  I                                                231
             GO TO 3200                                                  232
C-------------------------------------------------------------------
C     COMPUTE AND PRINT STATISTICS
 4000      CONTINUE                                                      233
           IF (LBNDRY .GT. 0)  GO TO 5000                                234
 4100      CALL SETIND(INDCCS)                                           235
           CALL TRKCHR(1H ,3)                                          * 236
           WRITE (LP,40)  NTIME,NCCS                                     237
           IF (NCCS .LE. 0)  GO TO 4400                                  238
           SDX  =  0.0                                                   239
           SDY  =  0.0                                                   240
           SDD  =  0.0                                                   241
           DO 4300  IC=1,NCCS                                            242
C            CALCULATE CLOSEST CLUSTER
             DMIN  =  1.0E10                                             243
             IMIN  =  0                                                  244
             DO 4200  I=1,NCCG                                           245
               DIST  =  (XCCG(I)-XCCS(IC))**2 + (YCCG(I)-YCCS(IC))**2    246
               IF (DIST .GE. DMIN)  GO TO 4200                           247
               DMIN  =  DIST                                             248
               IMIN  =  I                                                249
 4200          CONTINUE                                                  250
             DX   =  XCCG(IMIN) - XCCS(IC)                               251
             DY   =  YCCG(IMIN) - YCCS(IC)                               252
             DIST =  SQRT(DMIN)                                          253
             WRITE (LP,41)  IC,XCCS(IC),YCCS(IC)                         254
     *                      ,IMIN,XCCG(IMIN),YCCG(IMIN),DX,DY,DIST
             SDX  =  SDX + DX                                            255
             SDY  =  SDY + DY                                            256
             SDD  =  SDD + DIST                                          257
 4300      CONTINUE                                                      258
           R       =  1.0 / FLOAT(NCCS)                                  259
           SDX     =  SDX * R                                            260
           SDY     =  SDY * R                                            261
           SDD     =  SDD * R                                            262
           SUMDX   =  SUMDX + SDX                                        263
           SUMDY   =  SUMDY + SDY                                        264
           SUMDD   =  SUMDD + SDD                                        265
           SSQDX   =  SSQDX + SDX*SDX                                    266
           SSQDY   =  SSQDY + SDY*SDY                                    267
           SSQDD   =  SSQDD + SDD*SDD                                    268
           NTIME   =  NTIME + 1                                          269
 4400      CONTINUE                                                      270
           CALL PPLOT(PRTPLT,0.0,2)                                      271
C          SET SUBJECT CC#S IN PRINT PLOT TO BLANKS
           DO 4500  IC=1,NCCS                                            272
             IH  =  XCCS(IC)                                             273
             IV  =  YCCS(IC)                                             274
 4500      CALL PPLOT(PRTPLT,IH,IV,1H )                                * 275
           WRITE (LP,1)                                                  276
           GO TO 1005                                                    277
C-------------------------------------------------------------------
C     FORCE SUBJECT TO DRAW BOUNDARY
 5000      CALL BNDRY(PRTPLT)                                            278
           GO TO 4100                                                    279
C-------------------------------------------------------------------
C     EXIT TO MONITOR
 9000      CONTINUE                                                      280
```

```
          IF (NTIME .LE. 1)  GO TO 9100                                    281
          R  =  1.0/FLOAT(NTIME  )                                         282
          S  =  1.0/FLOAT(NTIME-1)                                         283
          SSQDX  =  SQRT((SSQDX - SUMDX*SUMDX*R)*S)                        284
          SSQDY  =  SQRT((SSQDY - SUMDY*SUMDY*R)*S)                        285
          SSQDD  =  SQRT((SSQDD - SUMDD*SUMDD*R)*S)                        286
          SUMDX  =  SUMDX * R                                              287
          SUMDY  =  SUMDY * R                                              288
          SUMDD  =  SUMDD * R                                              289
          WRITE (LP,20)                                                    290
          WRITE (LP,21)  NTIME,SUMDX,SUMDY,SUMDD,SSQDX,SSQDY,SSQDD         291
 9100     CONTINUE                                                         292
          WRITE (KB,22)  IRAND1,IRAND2                                     293
          WRITE (LP,22)  IRAND1,IRAND2                                     294
      END                                                                  295
      SUBROUTINE BNDRY(PRTPLT)                                              1
      DIMENSION PRTPLT(2)                                                   2
          RETURN                                                            3
      END                                                                   4
      SUBROUTINE RDTXT(N)                                                   1
C         MAY 18, 1972
C   READ A LINE OF TEXT
      COMMON /TXTBUF/  LUNTXT,MAXTXT,ICR,ITXTBF(80)                        2
      COMMON /LUNS/  KB,LP,KR                                               3
    1     FORMAT (80A1)                                                   4
    2     FORMAT (1X,80A1)                                                5
C    READ A LINE
          READ (LUNTXT,1,END=10)  (ITXTBF(I),I=1,MAXTXT)                   6
          WRITE (LP   ,2)  (ITXTBF(I),I=1,MAXTXT)                          7
          N  =  0                                                          8
          RETURN                                                           9
C    EOF ON DISK FILE -- FORCE SWITCH BACK TO TTY
   10     N  =  0                                                         10
          ITXTBF(1)  =  1HS                                             *  11
          RETURN                                                          12
      END                                                                 13
      SUBROUTINE INIT(INDBOX,IDATE)                                        1
C   INITIALIZE KEY PARAMETERS FOR PROGRAM PLACE
C          APRIL 26, 1972
C-------------------------------------------------------------------
      COMMON /BOUNDS/  MINH,MAXH,MINV,MAXV,HLEN,VLEN,IDOT                  2
      COMMON /DEBUG/  LDEBUG                                               3
      COMMON /LUNS/  KB,LP,KR                                              4
      COMMON /RANDS/  IRAND1,IRAND2                                        5
      COMMON /TXTBUF/  LUNTXT,MAXTXT,ICR,ITXTBF(80)                        6
      COMMON /VGD1/  INDISP,MAXDIS,DISBUF(3000)                            7
C-------------------------------------------------------------------
C     LOGICAL UNIT NUMBERS
          LP  =  5                                                         8
          KB  =  6                                                         9
          KR  =  1                                                        10
C    INITIAL TEXT WILL BE FROM TELETYPE
          LUNTXT  =  KB                                                   11
          MAXTXT  =  72                                                   12
          CALL SETFIL(KR,4HDATA)                                        *  13
C    RANDOM NUMBER GENERATOR
          IRAND1  =  0                                                    14
          IRAND2  =  0                                                    15
C    DISPLAY
          IDOT  =  1H.                                                    16
          MINH  =  -2047                                                  17
          MINV  =  -2047                                                  18
```

```
          MAXH  =   2047                                                    19
          MAXV  =   2047                                                    20
          HLEN  =  MAXH - MINH                                              21
          VLEN  =  MAXV - MINV                                              22
          MAXDIS  =  6000                                                   23
          CALL DPON                                                         24
C         A LARGE BLANK BUGMARK
          CALL TRKCHR(1H ,3)                                                *
C         BOX AROUND SCREEN                                                 25
          CALL MOVBM(MINH,MINV)
          CALL IVECT(MINH,MAXV,0)                                           26
          CALL IVECT(MAXH,MAXV,0)                                           27
          CALL IVECT(MAXH,MINV,0)                                           28
          CALL IVECT(MINH,MINV,0)                                           29
          CALL GETIND(INDBOX)                                               30
C    MISCELLANEOUS                                                          31
          CALL DATE(IDATE)
          LDEBUG  =  103                                                    32
          LDEBUG  =  0                                                      33
          RETURN                                                            34
      END                                                                   35
      SUBROUTINE DATE(N)                                                    36
C         MARCH 3, 1972                                                     1
      DIMENSION N(3)
C  DUMMY DATE ROUTINE AS IT IS NOT IN LIBRARY 16 FEB 72                     2
      DATA  I00/2H00/
          N(1)  =  I00                                                      3
          N(2)  =  I00                                                      4
          N(3)  =  I00                                                      5
          RETURN                                                            6
      END                                                                   7
      FUNCTION GETVAL(IBUF,N,OLD,ICR,IERR)                                  8
C                                                                           1
C         MARCH 7, 1972
C  COMPUTE A VALUE FROM THE INPUT TEXT CONTAINED IN BUFFER IBUF
C  ICR  END OF LINE INDICATOR
C  N    COUNTER -- NUMBER OF CHARACTERS USED IN INPUT TEXT
C  OLD  CURRENT VALUE OF THE PARAMETER
C  IERR ERROR RETURN -- 0=NO ERROR   1=END OF LINE (AFTER VALUE)
C                       2=ILLEGAL CHARACTER  3=END OF TEXT BEFORE VALUE
C-----------------------------------------------------------------------
      DIMENSION IBUF(2)
      DATA  IBCD0/1H0/, MAXN/72/                                            2
C                                                                           3
C    INITIALIZE
          NDAD    =  0
          SIGN    =  1.0                                                    4
          IDOT    =  0                                                      5
          NCH     =  0                                                      6
          IERR    =  0                                                      7
          VALUE   =  0.0                                                    8
          GETVAL  =  OLD                                                    9
  100     N  =  N + 1                                                       10
          IF (N .GT. MAXN)  GO TO 140                                       11
          JJ  =  IBUF(N)                                                    12
          IF (JJ .EQ. ICR)  GO TO 140                                       13
          IF (JJ .EQ. 1H )  IF (NCH)  100,100,200                           14
          IF (JJ .EQ. 1H,)  IF (NCH)  100,100,200                           15
          IF (JJ .GE. 1H0 .AND. JJ .LE. 1H9)  GO TO 110                     16
          IF (JJ .EQ. 1H+)  GO TO 100                                       17
          IF (JJ .EQ. 1H-)  GO TO 120                                       18
          IF (JJ .EQ. 1H.)  GO TO 130                                       19
                                                                            20
```

```
C          ILLEGAL CHARACTER                                                    21
           IERR  =  2                                                           22
           RETURN
C     NUMBER
  110      VALUE   =  10.0*VALUE  + (JJ-IBCD0)                                  23
           IF (IDOT .GT. 0)  NDAD = NDAD + 1                                    24
           NCH  =  NCH + 1                                                      25
           GO TO 100                                                            26
C    MINUS SIGN
  120      SIGN  =  -1.0                                                        27
           GO TO 100                                                            28
C  DECIMAL POINT
  130      IDOT  =  N                                                           29
           GO TO 100                                                            30
C  END OF TEXT
  140      IERR  =  3                                                           31
           IF (NCH .EQ. 0)  RETURN                                              32
           IERR  =  1                                                           33
C  END OF ROUTINE
  200      GETVAL  =  SIGN*VALUE/(10.0**NDAD)                                   34
           RETURN                                                               35
       END                                                                      36
       SUBROUTINE GENCC(ICC,PRTPLT)                                              1
C          MAY 18, 1972
C  GENERATE A CLUSTER WITH CENTER (XCCG(ICC),YCCG(ICC)) AND DEVIATIONS
C  (XDVG(ICC),YDVG(ICC)) CONTAINING NPTG(ICC) POINTS, RANDOM NORMAL
C  ABOUT THE CENTER.
C
       DIMENSION PRTPLT(2)                                                       2
       COMMON /DEBUG/  LDEBUG                                                    3
       COMMON /BOUNDS/  MINH,MAXH,MINV,MAXV,HLEN,VLEN,IDOT                      4
       COMMON  /STATS/  NCCO                                                     5
      1          ,NPTO(50),XCCO(50),YCCO(50),XDVO(50),YDVO(50)
      2          ,NPTG(50),XCCG(50),YCCG(50),XDVG(50),YDVG(50),COR(50)
       DATA  ICHA / 1HA /                                                        6
C
            SX    =  0.0                                                       7
            SY    =  0.0                                                       8
            SXX   =  0.0                                                       9
            SXY   =  0.0                                                      10
            SYY   =  0.0                                                      11
            N     =  0                                                        12
            NPT   =  NPTO(ICC)                                                13
            XCC   =  XCCO(ICC)                                                14
            YCC   =  YCCO(ICC)                                                15
            XDV   =  XDVO(ICC)                                                16
            YDV   =  YDVO(ICC)                                                17
            IF (LDEBUG.NE.0)  PRINT 2, ICC,NPT,XCC,YCC,XDV,YDV                18
     2      FORMAT (#0GENCC#,I3,I4,2F7.0,2X,2F6.0)                            19
            DO 10  I=1,NPT                                                    20
              CALL NORMAL(X)                                                  21
              XX  =  X*XDV + XCC                                              22
              IH  =  XX                                                       23
              IF (LDEBUG .GT. 1)  PRINT 3,I,IH,XX,X                           24
     3        FORMAT (5X,#GENCC - X#,I3,I6,F8.1,F10.6)                        25
              IF (IH.LE.MINH .OR .IH.GE.MAXH)  GO TO 10                       26
              CALL NORMAL(Y)                                                  27
              YY  =  Y*YDV + YCC                                              28
              IV  =  YY                                                       29
              IF (LDEBUG .GT. 1)  PRINT 4,I,IV,YY,Y                           30
     4        FORMAT (13X,#Y#,I3,I6,F8.1,F10.6)                               31
              IF (IV.LE.MINV .OR. IV.GE.MAXV)  GO TO 10                       32
```

```
C           DISPLAY THE POINT                                                    33
            N    =  N + 1                                                        34
            CALL IVECT(IH,IV,3)
C           CALL MOVBM(IH,IV)
C           CALL CHARS(IDOT,1,0)
C           COMPUTE SUMS FOR STATISTICS
            SX   =  SX + XX                                                      35
            SY   =  SY + YY                                                      36
            SXX  =  SXX + XX*XX                                                  37
            SYY  =  SYY + YY*YY                                                  38
            SXY  =  SXY + XX*YY                                                  39
            IF (LDEBUG .GE. 1)  PRINT 5, I,N,SX,SY,SXX,SYY,SXY                  40
    5       FORMAT (20X,2I4,5F10.0)                                              41
            CALL PPLOT(PRTPLT,IH,IV,IDOT)                                        42
   10     CONTINUE                                                              43
          IF (N .LE. 1)  RETURN                                                  44
          S  =  N - 1                                                            45
          R  =  1.0/FLOAT(N)                                                     46
          NPTG(ICC)  =  N                                                        47
          XDVG(ICC)  =  SQRT((SXX - SX*SX*R)/S)                                  48
          YDVG(ICC)  =  SQRT((SYY - SY*SY*R)/S)                                  49
          XCCG(ICC)  =  SX*R                                                     50
          YCCG(ICC)  =  SY*R                                                     51
          COR (ICC)  =  (SXY*R-XCCG(ICC)*YCCG(ICC))/(XDVG(ICC)*YDVG(ICC))        52
          IH   =  XCCG(ICC) + 0.5                                                53
          IV   =  YCCG(ICC) + 0.5                                                54
          ICH  =  ICHA + ICC - 1                                                 55
          CALL PPLOT(PRTPLT,IH,IV,ICH)                                           56
          RETURN                                                                 57
      END                                                                        58
      FUNCTION INDTXT(ITXTBF,NN)                                                  1
C         MARCH 3, 1972
C  FIND THE LAST NON-BLANK CHARACTER IN THE TEXT BUFFER ITXTBF
      DIMENSION ITXTBF(2)                                                        2
      DATA  IBL/1H /                                                             3
          DO 10  I=1,NN                                                          4
            N  =  NN + 1 - I                                                     5
            IF (ITXTBF(N) .NE. IBL)  GO TO 20                                    6
   10     CONTINUE                                                              7
   20     INDTXT  =  N                                                          8
          RETURN                                                                 9
      END                                                                        10
      SUBROUTINE NORMAL(R)                                                        1
C         APRIL 8, 1972
C   THIS METHOD OF GENERATING PSEUDO-RANDOM NORMAL(0,1) VARIATES CAME
C  FROM MARTIN, FRANCIS F., COMPUTER MODELING AND SIMULATION, WILEY AND
C  SONS, 1968, PAGE 80.
      COMMON /RANDS/  I1,I2                                                      2
          R  =  0.0                                                              3
          DO 10  I=1,7                                                           4
            CALL RANDU(I1,I2,RAND)                                               5
   10     R  =  R + RAND                                                        6
C         1.309307  = 1.0/SQRT( 7/12)
          R  =  (R-3.5)*1.309307                                                 7
          RETURN                                                                 8
C         DO 10  I=1,10
C         R  =  (R-5.0)*1.0954452
C         1.0954452 = 1.0/SQRT(10/12)
      END                                                                        9
      SUBROUTINE PPLOT(BUF,IH,IV,ICH)                                             1
C         APRIL 26, 1972
C  PUT CHARACTER #ICH# IN PRINT PLOT BUFFER #BUF# AT (IH,IV)
```

```
C  ICH=1 IMPLIES INITIALIZE BUFFER  ICH=2 IMPLIES PRINT THE BUFFER
      DIMENSION BUF(1200)
      COMMON /LUNS/  KB,LP,KR
      COMMON /BOUNDS/  MINH,MAXH,MINV,MAXV,HLEN,VLEN,IDOT
      DATA  IBL/1H /, IAS/2H**/, ASTER/4H****/, BLANK/4H    /          *
C
           IF (ICH .EQ. 1)  GO TO 10
           IF (ICH .EQ. 2)  GO TO 20
C  STORE ICH IN THE PROPER POSITION
           JH  =  FLOAT(IH-MINH)*80.0/HLEN
           JV  = -FLOAT(IV-MINV)*60.0/VLEN + 60.0
           JJ  =  JH + JV*80 -79
           CALL SETCH(BUF,JJ,ICH)
           RETURN
C  INITIALIZE THE BUFFER
   10      DO 11  I=81,4720
   11      CALL SETCH(BUF,I,IBL)
           DO 12  I=1,80
             CALL SETCH(BUF,I   ,IAS)
   12      CALL SETCH(BUF,I+4720,IAS)
           DO 13  I=1,60
             CALL SETCH(BUF,80*I ,IAS)
   13      CALL SETCH(BUF,80*I-79,IAS)
           RETURN
C  PRINT THE BUFFER
   20      WRITE (LP,21)
           WRITE (LP,22)  BUF
           WRITE (LP,22)
   21      FORMAT (#1PRINT PLOT OF DATA,  .=DATA   1=CC1, ETC#
     1           ,17H   *=SUBJECT CC#S )
   22      FORMAT (1X,20A4)
           RETURN
      END
      SUBROUTINE INTGR(IVAL,IBUF)
C          APRIL 12, 1972
C  ENCODE 4 DIGITS OF INTEGER VALUE, IVAL, WITH SIGN INTO 6 CHARACTERS
C  OF THE BUFFER IBUF.
      DIMENSION IBUF(3),NUMBER(100),IUNIT(10),NUMB1(50),NUMB2(50)
      EQUIVALENCE (NUMBER(1),NUMB1),(NUMBER(51),NUMB2)
      DATA IUNIT  /  2H 0,2H 1,2H 2,2H 3,2H 4,2H 5,2H 6,2H 7,2H 8,2H 9 /
      DATA  NUMB1  /
     +          2H00,2H01,2H02,2H03,2H04,2H05,2H06,2H07,2H08,2H09
     +         ,2H10,2H11,2H12,2H13,2H14,2H15,2H16,2H17,2H18,2H19
     +         ,2H20,2H21,2H22,2H23,2H24,2H25,2H26,2H27,2H28,2H29
     +         ,2H30,2H31,2H32,2H33,2H34,2H35,2H36,2H37,2H38,2H39
     +         ,2H40,2H41,2H42,2H43,2H44,2H45,2H46,2H47,2H48,2H49  /
      DATA  NUMB2  /
     +          2H50,2H51,2H52,2H53,2H54,2H55,2H56,2H57,2H58,2H59
     +         ,2H60,2H61,2H62,2H63,2H67,2H45,2H66,2H67,2H68,2H69
     +         ,2H70,2H71,2H72,2H73,2H74,2H75,2H76,2H77,2H78,2H79
     +         ,2H80,2H81,2H82,2H83,2H84,2H85,2H86,2H87,2H88,2H89
     +         ,2H90,2H91,2H92,2H93,2H94,2H95,2H96,2H97,2H98,2H99  /
C     INITIALIZE
           N99   =  1
           NUMB  =  IABS(IVAL)                                          *
           IBUF(1)  =  2H                                               *
           IBUF(2)  =  2H
           IBUF(3)  =  2H**
           IF (IVAL .LT. 0)  IBUF(1) = 2H -
           IF (NUMB .LE. 99)  GO TO 10
C    VALUE HAS MORE THAN 2 DIGITS
           N99   =  NUMB/100 + 1
```

```
      IF (N99 .GT. 100)  RETURN                                          15
      IBUF(2)  =  NUMBER(N99)                                            16
      IF (N99 .LE. 10)  IBUF(2) = IUNIT(N99)                             17
C  RIGHT-MOST TWO DIGITS
10    N99   =  NUMB - (N99-1)*100 + 1                                    18
      IBUF(3)  =  NUMBER(N99)                                            19
      IF (NUMB .LE. 10)  IBUF(3) = IUNIT(N99)                            20
      RETURN                                                             21
      END                                                                22
```

REFERENCES

1. Chow, C. K., "An Optimum Character Recognition System Using Decision Functions," IRE Trans. Elec. Comp., Vol. EC-6, pp. 247-253, December 1957.

2. Kanal, L. and Chandrasekaran, B., "Recognition, Machine Recognition, and Statistical Approaches," in Methodologies of Pattern Recognition," Academic Press, New York, New York, pp. 317-332, 1969.

3. Nagy, G., "State of the Art in Pattern Recognition," Proc. IEEE, Vol. 56, pp. 836-862, May 1968.

4. Ho, Y. C. and Agrawala, A., "On Pattern Classification Algorithms: Introduction and Survey," Proc. IEEE, Vol. 56, pp. 2101-2114, December 1968.

5. Bolshev, L. N., "Cluster Analysis," Bulletin, International Statistical Institute, Vol. 43, pp. 411-425, 1969.

6. Ball, G. H., "Data Analysis in the Social Sciences: What About the Details?," Proc. FJCC, Spartan Books, Washington, D. C., pp. 533-560, 1969.

7. Marill, T. and Green, D. M., "Statistical Recognition Functions and the Design of Pattern Recognizers," IRE Trans. Elec. Comp., Vol. EC-9, pp. 472-477, December 1960.

8. Minsky, M., "Steps Toward Artificial Intelligence," Proc. IRE, Vol. 49, pp. 8-30, January 1961.

9. Bahadur, R. R., "A Representation of the Joint Distribution of Responses to n Dichotomous Items," in Studies in Item Analysis and Prediction, edited by H. Solomon, Stanford University Press, Stanford-California, pp. 158-168, 1961.

10. Chow, C. K., "A Class of Nonlinear Recognition Procedures," IEEE Trans. Sys. Sci. Cyb., Vol. SSC-2, pp. 101-109, December 1966.

11 Chow, C. K., and Liu, C. N., "Approximating Discrete Probability Distributions with Dependence Trees," IEEE Trans. Info. Theory, Vol. II-14, pp. 462-467, May 1968.

12. Abramson, N. and Braverman, D., "Learning to Recognize Patterns in a Random Environment," IRE Trans. Info. Theory, Vol. IT-8, pp. S58-S63, September 1962.

13 Geisser, S., "Predictive Discrimination," in Multivariate Analysis, edited by P. R. Krishnaiah, Academic Press, New York, New York, pp. 149-163, 1966.

14. Kanal, L. N. and Chandrasekaran, B., "On Dimensionality and Sample Size in Statistical Pattern Recognition," Proc. NEC, Vol. 24, pp. 2-7, 1968.

15. Meisel, W. S., Computer-Oriented Approaches to Pattern Recognition, Academic Press, New York, New York, 1972.

16. Fix, E. and Hodges, J. L., Jr., "Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties," Report No. 4, USAF School of Aviation Medicine, Randolph Field, Texas, February 1951.

17 Sebestyen, G. and Edie, J., "An Algorithm for Non-Parametric Pattern Recognition," IEEE Trans. Elec. Comp., Vol. EC-15, pp. 908-915, December 1966.

18. Cover, T. M. and Hart, P. E., "Nearest Neighbor Pattern Classification," IEEE Trans. Info. Theory, Vol. IT-13, pp. 21-27, January 1967.

19. Wilson, D. L., " Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," IEEE Trans. Sys., Man, Cyb., Vol. SMC-2, pp. 408-421, July 1972.

20. Gibbons, J. D., Nonparametric Statistical Inference, McGraw-Hill Book Company, New York, New York, 1971.

21. Fisher, R. A., "The Use of Multiple Measurements in Taxonomic Problems," Annals of Eugenics, Vol. 7, Part 2, pp. 179-188, 1936. Also in Contributions to Mathematical Statistics, John Wiley and Sons, Incorporated, New York, New York, 1963.

22. Rosenblatt, F., Principles of Neurodynamics, Spartan Books, Washington, D. C., 1962

23. Highleyman, W. H., "Linear Decision Functions, with Application to Pattern Recognition," Proc. IRE, Vol. 50, pp. 1501-1514, June 1962.

24. Brain. A. E. et al., "A Large, Self-Contained Learning Machine," Paper 6.1, WESCON, San Francisco, California, August 1963.

25. Duda, R. O. and Hart, P. E., Pattern Classification and Scene Analysis, John Wiley & Sons, Incorporated, New York, New York, 1972.

26. Ball, G. H., "Classification Analysis," Technical Note, Contract Nonr. 4918(00), SRI Project 5533, Stanford Research Institute, Menlo Park, California, November 1970.

27. Ling, R. F., "Cluster Analysis," Technical Report No. 18, Department of Statistics, Yale University, New Haven, Connecticut, January 1971.

28. Dorofeyuk, A. A., "Automatic Classification Algorithms, (Review)," Automation and Remote Control, Vol. 32, pp. 1928-1958, December 1971.

29. Spragins, J., "Learning Without a Teacher," IEEE Trans. Info. Theory, Vol. IT-12, pp. 223-230, April 1966.

30. Cooper, P. W., "Non-supervised Learning in Adaptive Statistical Pattern Recognition," Proc. 1968 IFIP Congress, Booklet J, pp. J71-J76, August 1968.

31. _____, "Nonsupervised Learning in Statistical Pattern Recognition," in Methodologies of Pattern Recognition, edited by S. Watanabe, Academic Press, New York, New York, pp. 97-109, 1969.

32. Williams, W. T. and Dale, M. B., "Fundamental Problems in Numerical Taxonomy," Advances in Botanical Research, Vol. 2, pp. 35-68, July 1964.

33. Sneath, P.H.A., "Evaluation of Clustering Methods," in Numerical Taxonomy, edited by A. J. Cole, Academic Press, New York, New York, pp. 257-271, 1969.

34. Wishart, D., "Mode Analysis: A Generalization of Nearest Neighbor Which Reduces Chaining Effects," in Numerical Taxonomy, edited by A. J. Cole, Academic Press, New York, New York, pp. 282-308, 1969.

35. Sokal, R. R. and Sneath, P.H.A., Principles of Numerical Taxonomy, W. H. Freeman, San Francisco, California, 1963.

36. Jardine, N. and Sibson, R., Mathematical Taxonomy, John Wiley & Sons, Incorporated, London, England, 1971.

37. Tryon, R. C. and Bailey, D. E., Cluster Analysis, McGraw-Hill Book Company, New York, New York, 1970.

38. Watanabe, M. S., Knowing and Guessing, John Wiley & Sons, Incorporated, New York, New York, 1969.

39. Fleiss, J. L. and Zubin, J., "On the Methods and Theory of Clustering," Multivariate Behavioral Research, Vol. 4, pp. 235-250, April 1969.

40. Pearson, K., "Contributions to the Mathematical Theory of Evolution," Philosophical Transactions of the Royal Society of London, Vol. 185, pp. 71-110, 1894.

41. Day, N. E., "Estimating the Components of a Mixture of Normal Distributions," Biometrika, Vol. 56, pp. 463-474, December 1969.

42. Wolfe, J. H., "A Computer Program for the Maximum-Likelihood Analysis of Types," Technical Bulletin 65-15, U. S. Naval Personnel Research Activity, San Diego, California, 1965.

43. _____, "Pattern Clustering by Multivariate Mixture Analysis," Research Memorandum SRM 69-17, AD684087, U. S. Naval Personnel Research Activity, San Diego, California, March 1969.

44. _____, "Pattern Clustering by Multivariate Mixture Analysis," Multivariate Behavioral Research, Vol. 5, pp. 329-350, July 1970.

45. _____, "NORMIX Program Documentation," Research Memorandum SRM 69-11, AD682213, U. S. Naval Personnel Research Activity, San Diego, California, December 1968.

46. Sebestyen, G. S., "Pattern Recognition by an Adaptive Process of Sample Set Construction," IRE Trans. Info. Theory, Vol. IT-8, pp. S82-S91, September 1962.

47. Ball, G. H. and Hall, D. J., "ISODATA, A Novel Method of Data Analysis and Pattern Classification," AD699616, Technical Report, Stanford Research Institute, Menlo Park, California, 1965.

48. Fortier, J. J. and Solomon, H., "Clustering Procedures," in Multivariate Analysis, edited by P. R. Krishnaiah, Academic Press, New York, New York, pp. 493-519, 1966.

49. Friedman, H. P. and Rubin, J., "On Some Invariant Criteria for Grouping Data," J. ASA, Vol. 62, pp. 1159-1178, December 1967.

50. Hartigan, J. A., "Representation of Similarity Matrices by Trees," J. ASA, Vol. 62, pp. 1140-1158, December 1967.

51. Gower, J. C. and Ross, G. J. S., "Minimum Spanning Trees and Single Linkage Cluster Analysis," Appl. Statistics, Vol.18, pp. 54-64, 1969.

52. Zahn, C. T., "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," IEEE Trans. Comp., Vol. C-20, pp. 68-86, January 1971.

53. Stevens, M. E., "Research and Development in the Computer and Information Sciences. 1. Information Acquisition, Sensing, and Input: A Selective Literature Review," National Bureau of Standards Monograph 113, Vol. 1, March 1970.

54. Chodrow, M. M., Bivona, W. A., and Walsh, G. M., "A Study of Handprinted Character Recognition Techniques," Final Report No. RADC-TR-65-444, AD479049, Information Processing Branch, Rome Air Development Center, Griffiss Air Force Base, New York, February 1966.

55. Lindgren, N., "Machine Recognition of Human Language--Part I," IEEE Spectrum, Vol. 2, pp. 114-136, March 1965; "Part II," pp. 44-59, April 1965; "Part III," pp.104-116, May 1965.

56. Andersson, P. L., "Optical Character Recognition--A Survey," Datamation, pp. 43-48, July 1969.

57. Andrews, D. R., Atrubin, A. J., and Hu, K. C., "The IBM 1975 Optical Page Reader. Part III: Recognition Logic Development," IBM Journal, Vol. 12, pp. 364-371, September 1968.

58. Casey, R. G. and Nagy, G., "An Autonomous Reading Machine," IEEE Trans. Comp., Vol. C-17, pp. 492-503, May 1968.

59. Munson, J. H., "Experiements in the Recognition of Handprinted Text: Part I--Character Recognition," Proc. FJCC, pp. 1125-1138, December 1968.

60. Groner, G., "Real-Time Recognition of Hand-Printed Text," Proc. FJCC, pp. 591-601, November 1966.

61. Freeman, H., "A Review of Relevant Problems in the Processing of Line-Drawing Data," in Automatic Interpretation and Classification of Images, edited by A. Grasselli, Academic Press, New York, New York, pp. 155-174, 1969.

62. Wegstein, J. H., Rafferty, J. F., and Pencak, W. J., "Matching Fingerprints by Computer," Technical Note 466, National Bureau of Standards, July 1968.

63. Swoboda, W. and Gerdes, J. W., "A System for Demonstrating the Effects of Changing Background on Automatic Target Recognition," in Pictorial Pattern Recognition, edited by G. C. Cheng et al., Thompson Book Company, Washington, D. C., pp. 33-43, 1968.

64. Harley, T. J., Kanal, L. N., and Randall, N. C., "Systems Considerations for Automatic Imagery Screening," In Pictorial Pattern Recognition, edited by G. C. Cheng et al., Thompson Book Company, Washington, D. C., pp. 15-32, 1968.

65. Hawkins, J. K., "Image Processing: A Review and Projection," in Automatic Interpretation and Classification of Images, edited by A. Grasselli, Academic Press, New York, New York, pp. 199-234, 1969.

66. Mendelsohn, M. and Prewitt, J., "Analysis of Cell Images," Annals of the New York Academy of Sciences, Vol. 128, pp. 1035-1053, 1966.

67. Ledley, R. S. and Ruddle, F. H., "Chromosome Analysis by Computer," Scientific American, Vol. 214, pp. 40-46, April 1966.

68. Dwyer, S. J. et al., "Computer Diagnosis of Radiographic Images," Proc. SFCC, Vol. 40, pp. 1027-1041, 1972.

69. Middleton, D., An Introduction to Statistical Communication Theory, McGraw-Hill Book Company, New York, New York, 1960.

70. Helstrom, C., Statistical Theory of Signal Detection, Pergamon Press, Oxford, England, 1960.

71. Selin, I., Detection Theory, Princeton University Press, Princeton, New Jersey, 1965.

72. Van Trees, H. L., Detection, Estimation, and Modulation Theory, Part I, John Wiley & Sons, New York, New York, 1968.

73. Van Trees, H. L., Detection, Estimation, and Modulation Theory, Part II, John Wiley & Sons, New York, New York, 1971.

74. _____, Detection, Estimation and Modulation Theory, Part III, John Wiley & Sons, New York, New York, 1971.

75. Hecker, M. H. L., "Speaker Recognition: An Interpretive Survey of the Literature," ASHA Monograph No. 16, American Speech and Hearing Association, Washington, D. C., January 1971.

76. Hyde, S. R., "Automatic Speech Recognition: Literature Survey and Discussion," Research Department Report No. 35, P. O. Research Department, Dollis Hill, London, 1968.

77. Young, J. R. and Hecker, M. H. L., "Some Observations on the Problem of Machine Recognition of Speech," Proc. NEC, Vol. 24, pp. 23-28, 1968.

78. Hill, D. R., "Man-Machine Interaction Using Speech," In Advances in Computers, Vol. II, edited by F. L. Alt, M. Rubinoff, and M. C. Yovits, Academic Press, New York, New York, pp. 165-230, 1971.

79. Offen, K. W., "Approaches to the Machine Recognition of Conversational Speech," in Advances in Computers, Vol. II, edited by F. L. Alt, M. Rubinoff, and M. C. Yovits, Academic Press, New York, New York, pp. 127-163, 1971.

80. Hemdal, J. F. and Hughes, G. W., "A Feature Based Computer Recognition Program for the Modeling of Vowel Perception," in Models for the Perception of Speech and Visual Form, edited by W. Wathen-Dunn, MIT Press, Cambridge, Massachusetts, 1967.

81. Martin, T. B., Nelson, A. L., and Zadell, H. J., "Speech Recognition by Feature-Abstraction Techniques," Technical Documentary Report AL-TDR-64-176, AD605891, Air Force Avionics Laboratory, Wright-Patterson Air Force Base, Ohio, August 1964.

82. Halle, M. and Stevens, K. N., "Speech Recognition: A Model and a Program for Research," IRE Trans. Info. Theory, Vol. IT8, pp. 155-159, February 1962.

83. Vicens, P., "Aspects of Speech Recognition by Computer," Report CS-127, Ph.D. Thesis, Computer Science Department, Stanford University, Stanford, California, 1969.

84. Newell, A. et al., "Speech-Understanding Systems: Final Report of a Study Group," Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, May 1971.

85. Rao, C. R., Advanced Statistical Methods in Biometric Research, John Wiley & Sons, New York, New York, 1952.

86. Shepard, R. N. and Carroll, J. D., "Parametric Representation of Nonlinear Data Structures," in Multivariate Analysis, edited by P. R. Krishnaiah, Academic Press, New York, New York, pp. 561-592, 1966.

87. Hall, D. J., Tepping, and Ball, G. H., "Applications of Cluster Analysis to Bureau of the Census Data," Final Report, Contract Cco-9312, SRI Project 7600, Stanford Research Institute, Menlo Park, California, 1970.

88. Bonner, R. E., "On Some Clustering Techniques," IBM Journal, Vol. 8, pp. 22-32, January 1964.

89. Bargmann, R. E. and Garney, R., "An Algorithm for Identifying and Testing Virtual Clusters," Technical Report No. 42, AD703859, Department of Statistics, University of Georgia, Athens, Georgia, October 1969.

90. _____, "Tables of Significance Tests for Virtual Clusters," Technical Report No. 36, AD700902, Department of Statistics, University of Georgia, Athens, Georgia, October 1969.

91. Darling, D. A., "The Kolmogorov-Smirnov, Cramer-Von Mises Tests," Ann. Math. Stat., Vol. 28, pp. 823-838, 1957.

92. Birnbaum, Z. W., "Numerical Tabulation of the Distribution of Kolmogorov's Statistic for Finite Sample Size," J. ASA, Vol. 47, pp. 431ff, 1952.

93. Lilliefors, H. W., "On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown," J. ASA, Vol. 62, pp. 399-402 June 1967.

94. Simpson, P. B., "Note on the Estimation of a Bivariate Distribution Function," Ann. Math. Stat., Vol. 23, pp. 470-472, 1952.

95. Duda, R. O., Mancuso, R. L., and Paskert, P. F., "Analysis of Techniques for Describing the State of Sky Through Automation," Final Report No. FAA-RD-71-52, AD735213, Contract E-203-69(N), SRI Project 7935, Stanford Research Institute, Menlo Park, California, July 1971.

96. Darling, D. A., "The Cramer-Smirnov Test in the Parametric Case," Ann. Math. Stat., Vol. 26, pp. 1-20, March 1955.

97. Kruskal, J. B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," Proc. Amer. Math. Soc., Vol. 7, pp. 48-50, 1956.

98. Prim, R. C., "Shortest Connection Networks and Some Generalizations," Bell. Sys. Tech. J., Vol. 36, pp. 1389-1401. November 1957.

99. Dijkstra, E. W., "A Note on Two Problems in Connection with Graphs, Numer. Math., Vol. 5, pp. 269-271, October 1959.

100. Whitney, V. K. M., "Algorithm 422-Minimal Spanning Tree [H]," C. ACM, Vol. 15, pp. 273-274, April 1972.

101. Green, P. E. and Carmone, F. J., Multidimensional Scaling and Related Techniques in Marketing Analysis, Allyn and Bacon, Boston, 1970.

102. Torgerson, W. S., Theory and Methods of Scaling, John Wiley & Sons, New York, New York, 1958.

103. Krantz, D. H., "Measurement Structures and Psychological Laws," Science, Vol. 175, No. 31, pp. 1427-1435, March 1972.

104. Ball, G. H. and Hall, D. J., "Research on ISODATA Techniques," Final Report SRI Project 5533, Stanford Research Institute, Menlo Park, California, January 1972.

105. Hall, D. J., Ball. G. H., and Wolf, D. E., "Applications of the PROMENADE Data-Analysis System," Computers and Communcations Conference Record, IEEE, Mohawk Valley Section, pp. 101-108, 1969.

106. Einstein, Albert, The Meaning of Relativity, Princeton University Press, Princeton, New Jersey, 1921, fifth edition 1955.

107. Gabor, D., "Theory of Communication," Inst. of Electrical Engineers J., Vol. 93, Part III, pp. 429-457, 1946.

108. MacKay, D. M., "Quantal Aspects of Scientific Information," Phil. Mag., Ser. 7, Vol. 41, No. 314, pp. 289-311, March 1950.

109. Julesz, B., "Cluster Formation at Various Perceptual Levels," in "Methodologies of Pattern Recognition," by S. Watanabe, Proc. of the Int. Conf. on Methodologies of Pattern Recognition, Academic Press, New York, New York, pp. 297-315, 1969.

110 Eusebio, J. W. and Ball, G. H., "ISODATA-Lines--A Program for Describing Multivariate Data by Piecewise-Linear Curves," Proc. of the Int. Conf. on Systems Science and Cybernetics, University of Hawaii, Honolulu, Hawaii, pp. 560-563, 1968.

111. Burstall, R. M., Leaver, R. A., and Sussams, J. E., "Evaluation of Transport Costs for Alternative Factory Sites--A Case Study," Operational Research Quart., Vol. 13, No. 4, December 1962.

112. Parzen, E., "On Estimation of a Probability Density Function and Mode," Amer. Math. Stat., Vol. 33, pp. 1067-1076, 1962.

113. Bohm, D., "On the Role of Hidden Variables in the Fundamental Structure of Physics," Contemporary Physics, Vol. II, pp. 95-116, Trieste Symposium, 1970.

114. Born, M., Einstein's Theory of Relativity, revised edition, Dover Publications New York, New York, 1965.

115. Einstein, Albert, Essays in Science, Philosophical Library, New York, New York, 1934.

116. Mahalanobis, P. C., "On the Generalized Distance in Statistics," Proc. India National Inst. of Science, Vol. 2, pp. 49-55, Calcutta, India, 1936.

117. Mucciardi, A. N., "Information Filtering Using the CLUSTER Algorithm," Proc. of Computer Image Processing and Recognition, Vol. 2, pp. 24-26, August 1972.

118. Weinberg, G. M., The Psychology of Computer Programming, Van Nostrand, New York, New York, 1971.

119. Narasimhan, R., "Intelligence and Artificial Intelligence," in Computer Studies in the Humanities and Verbal Behavior, by F. R. Horowitz, Mouton and Company, P. O. Box 1132 The Hague, Netherlands, 1971.

120. Newell, A. and Simon, H., Human Problem Solving, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1972.

121. Perls, F., Gestalt Therapy Verbatim, Real People Press, Lafayette, California, 1969.

122. Hall, D. J., "Man-Machine Projects at SRI," Int. J. Man-Machine Studies, Vol. 2, pp. 363-394, 1970.

123. Sackman, H. and Citrenbaum, R. L., Online Planning Towards Creative Problem-Solving, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1972.

124. Hall, D. J. et al., "PROMENADE--An Interactive Graphics Pattern-Recognition System," Proc. of the IFIP Congress 68, pp. 951-956, August 1968. Also IFIP Congress 68, Final Supplement, Booklet J, pp. J46-J50, August 1968.

125. Hall, D. J., Ball, G. H., and Wolf, D. E., "Interactive Graphic Clustering Using the PROMENADE System," Proc. of the 1969 Soc. Stat. Section, Amer. Stat. Assn., pp. 65-73, 1969.

126. Yourdon, E., Design of On-Line Computer Systems, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1972.

127. Ball, G. H. and Hall, D. J., "Some Implications of Interactive Graphic Computer Systems for Data Analysis and Statistics," Technometrics, Vol. 12, No. 1, pp. 17-31, February 1970.

128. Watanabe, S., "Methodologies of Pattern Recognition," Proc. of the Int. Conf. on Methodologies of Pattern Recognition, Academic Press, New York, New York, 1969.

129. Bohm, D., "An Inquiry into the Function of Language and Thought," based on a colloquy held at the Institute of Contempory Arts, London, 28 March 1971 (personal communication).

130. Smith, D. E., "Laplace on the Probability of Errors in the Mean Results of a Great Number of Observations, Etc.," in A Source Book in Mathematics, Dover Publishing, New York, New York, pp. 588-604, 1959.

131. Fischler, M. A., "Machine Perception and Description of Pictorial Data," Proc. IJCAI, pp. 629-639, Lockheed Palo Alto Research Laboratory, Palo Alto, California, 1969.

132. Narasimhan, R., "Programming Languages and Computers: A Unified Metatheory," Computer Group, Tata Institute of Fundamental Research, Bombay, India; in Advances in Computers, Vol. 8, Academic Press, New York, New York, 1967.

133. Evans, S. H., "Parameters of Human Pattern Perception," Report No. 6, Contract No. DAAD05-68-C-0176, AD713185. Institute for the Study of Cognitive Systems, Texas Christian University, Fort Worth, Texas, October 1970.

134. Guilford, J. P., Psychometric Methods, McGraw-Hill Book Company, Incorporated, New York, New York, 1954.

135. Swets, J. N., Signal Detection and Recognition by Human Observers, John Wiley & Sons, New York, New York, 1964.

136. Gregory, R. L., "Visual Illusions," Scientific American, pp. 66-76, November 1968.

137. Attneave, F., "Multistability in Perception," Scientific American, pp. 63-71, December 1971.

138. Ball, G. H. and Hall, D. J., "PROMENADE, An On-Line Pattern Recognition System," Final Technical Report No. RADC-TR-67-310, AD822174, Contract No. AF 30(602)-4196, SRI Project 6004, Stanford Research Institute, Menlo Park, California, September 1967.

139. Rand, W. M., "Objective Criteria for the Evaluation of Clustering Methods," J. of the Amer. Stat. Assn., Vol. 66, No. 336, pp. 846-850, December 1971.

140. Tiao, G. C. and Box, G. E. P., "Some Comments on Bayes' estimators," Tecnical Report 297, University of Wisconsin, Department of Statistics, Madison, Wisconsin, April 1972.

141. Endlich, R. M. et al., "Use of a Pattern Recognition Technique for Determining Cloud Motions from Sequences of Satellite Photographs," J. Appl. Meteor, Vol. 10, No. 1, pp. 104-117, 1971.

142. Hall, D. J. et al., "Objective Methods for Registering Landmarks and Determining Cloud Motions from Satellite Data," IEEE Trans. on Comput., Vol. C-21, No. 7, p. 768, July 1972.

143. Serebreny, S. M. et al., "Electronic System for Utilization of Satellite Cloud Pictures," Bull. Amer. Meteor, Soc., Vol. 51, No. 9, pp. 848-855, 1970.

144. Wolf, D. E., Endlich, R. M., and Hall, D. J., "Further Development of Objective Methods for Registering Landmarks and Determining Cloud Motions from Satellite Data," SRI Project 1005, Stanford Research Institute, Menlo Park, California, September 1972.

145. Hall, D. J. et al., "PROMENADE--An Improved Interactive-Graphics Man/Machine System for Pattern Recognition," Final Technical Report No. RADC-TR-68-572, AD692752, Contract No. F30603-67-C-0351, SRI Project 6737, Stanford Research Institute, Menlo Park, California, June 1969.

146. Earley, J., "Toward an Understanding of Data Structures," Comm. of ACM, Vol. 14, pp. 617-627, October 1971.