

Clustering Using a Similarity Measure Based on Shared Near Neighbors

R. A. JARVIS AND EDWARD A. PATRICK

Abstract—A nonparametric clustering technique incorporating the concept of similarity based on the sharing of near neighbors is presented. In addition to being an essentially parallel approach, the computational elegance of the method is such that the scheme is applicable to a wide class of practical problems involving large sample size and high dimensionality. No attempt is made to show how *a priori* problem knowledge can be introduced into the procedure.

Index Terms—Clustering, nonparametric, pattern recognition, shared near neighbors, similarity measure.

I. INTRODUCTION

WHEN speaking of clustering as a partitioning of data sets into subsets whose in-class members are “similar” in some sense and whose cross-class members are “dissimilar” in a corresponding sense, we are putting the whole weight of any computational method that reflects this loose definition on a suitable mathematical description of “similarity”. More often than not, this involves investigation in the choice of a metric, whether local, global, interactively modifiable [1]–[3], [5], probability density function family dependent [4], or a mixture of these. The whole matter of scaling, linear and nonlinear distortion, and search for invariance of measure under distortion circumstances demands attention. When developing clustering performance indices based on compactness measure, dispersion, and ratios of expected in-class compactness over expected cross-class dispersions [7]–[9], these indices usually reflect the investigator’s concept of what a cluster looks like. If it can be accepted that clusters are more or less “globular” in shape (see Fig. 1) so that the “cluster center” is a meaningful point to be associated with all the members of the cluster, then methods such as ISODATA [6] and Dorofeyuk’s technique [8] are very suitable and the techniques of Fukunaga and Koontz [9] (refining the approach in search for a criterion invariant under nonsingular linear transformation) are highly recommended.

In another work, Patrick and Fischer [1] find a property of the samples nearest a particular sample by using a test function. Particular samples with similar properties can be grouped as a cluster. Thus samples are similar only if their neighbors have similar properties where neighbors are located using a test

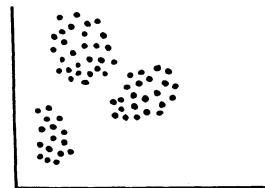


Fig. 1. Globular clusters.

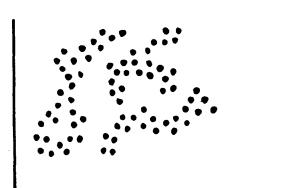


Fig. 2. Nonglobular clusters.

function. In the paper by Patrick and Fischer, Gaussian statistics are assumed such that appropriate properties are the mean vector and covariance matrix. The method can be computationally expensive as a large number of distance evaluations have to be made; this computational expense is offset somewhat because the parametric nature of the approach provides for the introduction of *a priori* problem knowledge.

In another paper, Patrick and Shen [2] present an example where *a priori* knowledge that the clusters being sought should have uniform density on a spherical domain is used. As a result, a sample is added to a growing cluster (one cluster grown at a time) only if it, together with its K most similar neighbors (similarity measure with respect to a non-Euclidean distance measure), do not violate the above-stated property of a uniform density on a spherical domain. Patrick and Shen concentrate on the interactive introduction of *a priori* problem knowledge into the clustering process.

This paper concentrates on the problem of how to cluster data in a nonparametric way when the globular concept of a cluster is not acceptable (see Fig. 2). Often, when the clusters are distinctly nonglobular, they can be made globular under suitable transformation [12]; but a generalized global transformation that simultaneously produces globular structure for all clusters is hard to find and probably even harder to compute. What is wanted, whenever possible, is a way of clustering in the original space with the use of a suitable measure of similarity. The method presented in this paper is superficially similar in some aspects to the minimal spanning tree clustering methods of Zahn [15], although it was developed quite independently. The two algorithms are in fact distinct; this point will be amplified at a later stage.

Section II develops the philosophy of the approach, while Section III goes on to discuss the computational aspects of the algorithm. Sections IV, V, and VI are devoted to the discussion of a number of variations of the basic approach under problem dependent circumstances. Examples of clustering of data usually considered difficult to handle are given in Section VII to illustrate the potential of the shared near neighbor approach. Finally, in Section VIII an overall assessment is given, applica-

Manuscript received January 18, 1971; revised January 16, 1973. This work was supported by the Rome Air Development Center, Griffiss Air Force Base, Rome, N. Y., under Contract F 30 602 68-C-0186.

R. A. Jarvis was with the School of Electrical Engineering, Purdue University, Lafayette, Ind. He is now with the Department of Statistics, School of General Studies, Australian National University, Canberra, Australia.

E. A. Patrick is with the School of Electrical Engineering, Purdue University, Lafayette, Ind.

tions suggested, and further investigation areas mentioned. A comparison is made between Zahn's minimal spanning tree method and our shared near neighbor approach in the concluding section of the paper.

II. SIMILARITY BY SHARING OF NEAR NEIGHBORS

Let $\{x_1, x_2, \dots, x_n\}$ be a set of data vectors in an L dimensional Euclidean vector space. It is required that these n points be broken into M groups (M unknown) where each group fulfills the investigator's requirement of a meaningful cluster.

Consider any two points of the set $\{x_1, x_2, \dots, x_n\}$, say x_i and x_j . The following statements reflect the dominating viewpoint presented in this paper.

1) If x_i and x_j are sufficiently "similar", then the probability that they come from the same cluster approaches unity. If then, a suitable point pair similarity measure can be found, a clustering technique is easily devised. The similarity measure presented in this paper can be stated as follows.

2) Data points are similar to the extent that they share the same near neighbors; in particular, two data points are similar to the extent that their respective k nearest neighbor lists match. In addition, for this similarity measure to be valid, it is required that the tested points themselves belong to the common neighborhood. This avoids the possibility of combining a small relatively isolated number of points with a high density group, as illustrated in Fig. 3.

Both point pair and group similarity can be measured this way. Be warned however, that group similarity measures based on this concept favor the globular structure if the groups are large. To avoid this, only point pair similarity measure is used in the clustering technique presented in this and the next section.

This pair point similarity measure has its own built-in automatic scaling, i.e., where points are widely spread, neighborhoods (volume just containing the k nearest neighbors) expand; where points are tightly positioned, the neighborhood shrinks (see Fig. 4). Thus the procedure does not depend on critical, globally fixed distance thresholds; yet there is interactive control through specifying k and the number of shared neighbors that we wish to regard as sufficient for purposes of grouping.

III. COMPUTATIONAL ASPECTS

The clustering algorithm using the above similarity concept is carried out in the following manner.

Step 1: For each point of the data set $\{x_1, x_2, \dots, x_n\}$, list the k nearest neighbors (e.g., using Euclidean distance measure) by order number $(1, 2, \dots, k)$. Regard each point as its own zeroth neighbor; in this way the first entry in each neighborhood table row is a label indicating which point the list belongs to. The significance of this will be seen later. Once the neighborhood lists have been tabulated, the raw data can be discarded. The rest of the computation can be entirely integer (see Fig. 5).

Step 2: Set up an integer label table of length n , with each entry initially set to the first entry of the corresponding neighborhood row (see Fig. 5).

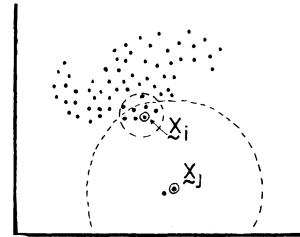


Fig. 3. Small group "trap" condition for $k = 7$; high proportion of shared neighbors.

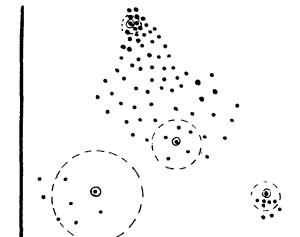


Fig. 4. Automatic scaling of neighborhoods (neighborhood including 5 nearest neighbors at points shown).

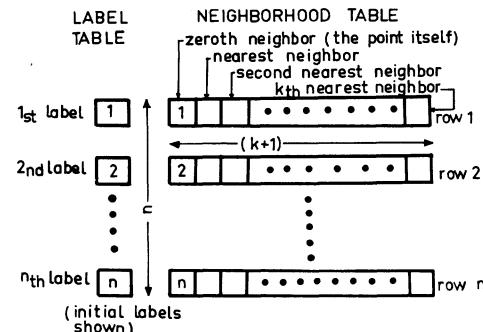


Fig. 5. Near neighbor and label lists (all entries are integer numbers).

Step 3: All possible pairs of neighborhood rows are tested in the following manner. Replace both label entries by the smaller of the two existing entries if both zeroth neighbors (the points being tested) are found in both neighborhood rows and at least k_t neighbor matches exist between the two rows (k_t is referred to as the similarity threshold). Also, replace all appearances of the higher label (throughout the entire label table) with the lower label if the above test is successful.

Step 4: The clusters under the k, k_t selections are now indicated by identical labeling of the points belonging to the clusters.

Step 5: Recomputation with new k and k_t can be carried out simply by returning to Step 2. (The first selection of k should be the largest the investigator will ever require so that the original vector data need not be recalled.)

Because Steps 2 and 3 are integer operations on a data set of size $n \times (k + 1)$, where k is usually $\ll n$ and may even be smaller than L , recalculation for pairs of (k, k_t) until the groupings are meaningful to the user is extremely fast and does not require the recall of the raw data. Propagating label changes as required in Step 3 can be made much faster than table search (especially for large n) by using a list linking algorithm that forms and utilizes chaining information about each group no matter how the members of the group are

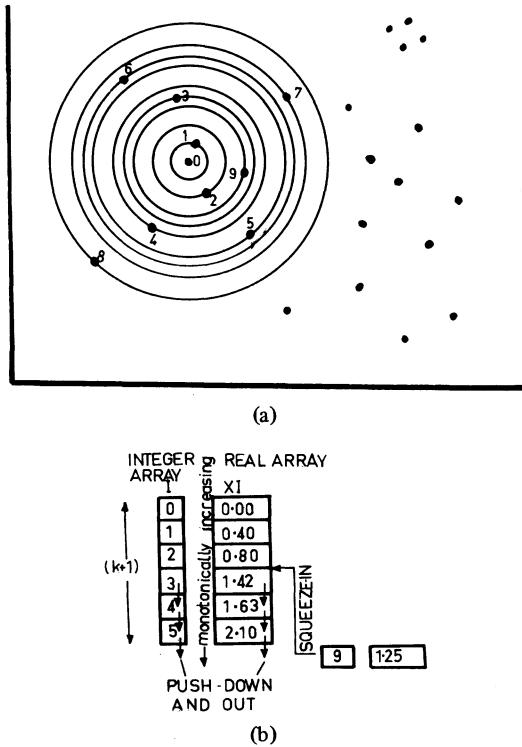


Fig. 6. Push-down and out/squeeze-in array example for generating the k nearest neighbors for a given sample point. (a) Sample points. (b) Push-down and out/squeeze-in array at an updating stage for $k = 5$.

scattered throughout the complete set of labels in the label table.

Although the Euclidean metric is mentioned in Step 1, the method is by no means restricted to this measure and any suitable measure can be used (not necessarily even a metric).

Even Step 1 is not as computationally expensive as it may appear. Two temporary data arrays I, XI , each of size $k + 1$, one integer and one real, are used for the generation of each neighborhood list row. For each neighborhood list row, initialize all I entries to zero and all XI entries to a very large number plus a number corresponding to the order in the array so that the last number is the largest and the numbers are monotonically increasing from the first entry to the $(k + 1)$ th entry. Only when a point in $\{x_1, \dots, x_n\}$ is closer (e.g., using Euclidean distance metric) than the last entry in XI (the largest entry) is the last entry pushed down out of the array and the new distance squeezed into the list to maintain monotonic order; the label of the new point replaces the corresponding I entry (see Fig. 6) value. When all points have been tested for the list, transfer I entries to the corresponding main (nontemporary) neighbor list array. Do this for all points in $\{x_1, x_2, \dots, x_n\}$.

All k nearest neighbors can be found with relatively little computational expense over that of finding the nearest neighbor by using the scheme previously described. For each neighborhood list row generation, only n distance measures need be made; when L is large this represents the largest computational expense of the algorithm.

The computational complexity of calculating the near neighbor table is of the order $(n)^2 L + C(k)$ operations where C is a relatively small factor to allow for the extra overhead of testing

for all k near neighbors for each point. Strictly speaking, only $n(n - 1)/2$ distance measures are necessary but in the algorithm implemented redundant distance calculations were tolerated to enhance programming simplicity and conserve storage. As these operations are on real (floating point) data they are relatively expensive, especially if the computer used has no floating point hardware. The computing complexity for one pass of the neighborhood table to explore clusters for neighborhood size k and "belongingness" threshold k_t is of the order of (at most) $(n(n - 1)/2)(k + 1)^2$ integer comparisons plus a data order and threshold dependent cost of the link listing procedure that is evoked only when matches of sufficient vote are detected. Considerable saving results in testing for the mutual appearance of the candidates in each other's near neighborhood and abandoning further consideration on failure of this test. Some actual times for typical data sets will be given later to illustrate the relative times for Step 1 and Step 3 of the above algorithm.

IV. WEIGHTED VOTING OF NEAR NEIGHBORS

A degree of refinement (at some computational expense) can be added to the clustering scheme by simply weighting the vote of shared neighbors according to their positions in the ordered neighborhood table row associated with each point. Since the ordering of the neighborhood tables arises naturally in the method of generation, the modification is trivial to implement. One need not necessarily simply count shared neighbors by an exhaustive matching check between members of each pair of the neighborhood tables and counting $N \leftarrow N + 1$, each time a match is found and compare this integer against a threshold. Rather, one could evaluate, for example, $N \leftarrow N + (k - r + 1) \times (k - q + 1)$ where r and q are the positions in the i th and j th neighborhood tables rows, respectively, and k is the total number of near neighbors evaluated. Then, if the nearest neighbors of both points are the same, the match gets $k \times k$ votes, while if the furthest neighbors (in the row of near neighbors) match, the match gets only 1 vote. In-between matches (vertical and diagonal) get vote weights between $k \times k$ and 1; all vote weights are integer. If the product weighting is too severe, then $N \leftarrow N + (k - r + 1) + (k - q + 1)$, or in fact any $N \leftarrow N + f(k, r, q)$, can be easily implemented (see Fig. 7).

This type of modification scheme is in keeping with the concept that the nearer the neighbor to a given point, the more information about the local density at the point that neighbor can provide; it is also related to the test function approach used in the cluster mapping procedure of Patrick and Fischer [1].

V. SIMILARITY MATRICES AND TREE STRUCTURES

If, after the calculation of the neighborhood tables for each of the n points of $\{x_1, x_2, \dots, x_n\}$, the investigator feels that the simple pair-by-pair chaining, according to a neighbor share threshold, is not adequately sophisticated to meaningfully cluster his data, he can, at extra computational expense, use one of at least two approaches.

Approach 1: He can use the pairwise similarity data derived from the neighborhood tables to construct a similarity matrix

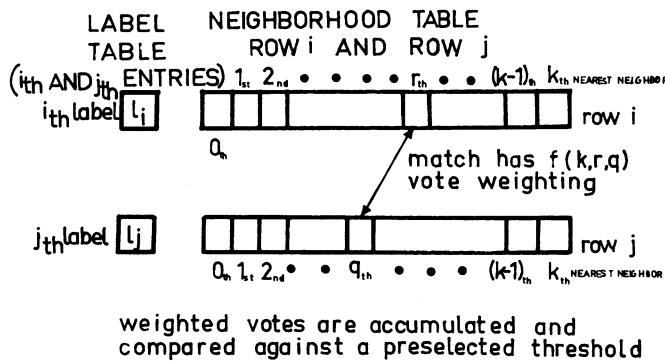
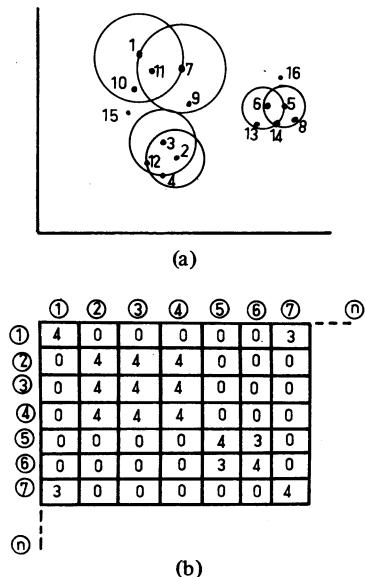


Fig. 7. Weighted voting of near neighbors.

Fig. 8. Example of similarity matrix using the number of shared near neighbors as point pair similarity measure with equally weighted votes. (a) Sample points. (b) Similarity matrix for $k = 3$.

and apply methods outlined by Bonner [13] or Gitman [11], which are performance index oriented. The similarity matrix, for a suitable value of $k < n$, will contain many zero entries, indicating the localized nature of the shared near neighbors similarity measure. (See Fig. 8.) Both the memory requirements ($n \times n$ similarity matrix) and the optimization of the selected criterion may be computationally prohibitive for large n .

Approach 2: He may cluster his data in hierarchical stages using the shared near neighbor or weighted vote shared near neighbor concept. First, he may link only very similar points by setting a high threshold value k_t . The number of groupings should then be larger than the approximately expected number of clusters. Then he could use a group-to-group similarity measure, still based on the near neighbor principle, and complete the next merge stage. He can continue this procedure until he is satisfied that the number of generated groups is in the range of his expectation. If all points in each pair of groups are used to measure group-to-group similarity, once again a "globular" cluster concept will be inferred at least in a piecewise sense (which may be a disadvantage). If it does seem disadvantageous to introduce this globular bias at this stage, a strict continuity type merge can be made by using

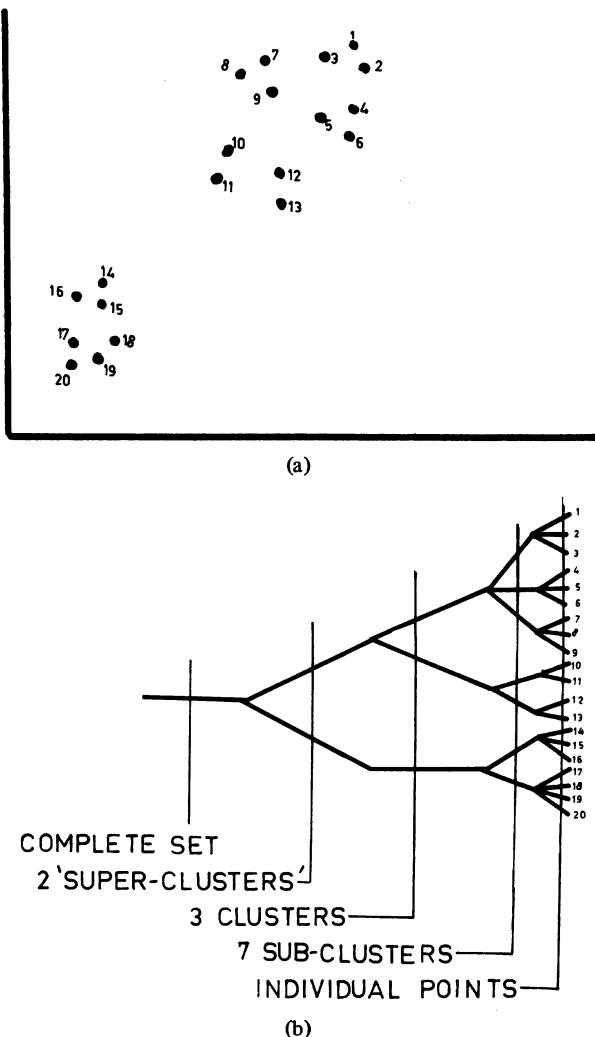


Fig. 9. Multilevel tree structure example. (a) Sample points. (b) Cluster tree.

only those points in both groups (of each group pair being tested) which have at least a certain number of near neighbors in the other group, and normalizing the measure by the number of such points. In this way, only the mutual borders of the groups will be effective in contributing towards merge decisions.

This multilayer approach has the advantage of analyzing the data at different levels of detail and this detail can be conveyed to the investigator by outputting the results in connective tree or dendrogram form [14]. This is extremely valuable if any single point or small group of points requires careful investigation at any later stage. That is, instead of generating one set of clusters, generate a family of sets of clusters, each member of the family at a different level of detail; the investigator can then select the level of interest at leisure (at a later time) without feeling that he may have missed something important by overgeneralization. (See example given in Fig. 9.)

VI. CLUSTERING BY COMBINED CONTINUITY AND GLOBULAR BIAS

If one does not want to restrict clusters to globular structures, then what makes Fig. 10(a) one cluster and Fig. 10(b) two clusters? It appears to be a matter of continuity rather

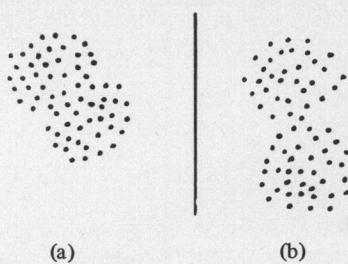


Fig. 10. Cluster examples. (a) Example. (b) Example.

than association or attraction to a center or mass that is involved here. The continuity measure should reflect a broad continuity of probability density function (pdf) across the portions of interest, taking the whole merging interface of the portions rather than a single or delicate bridging of the portions into consideration.

The use of the shared near neighbor rule should be interpreted with this concept of continuity in mind, i.e., the idea of an approximate or fuzzy continuity as the linking requirement in the forming of clusters.

Remember that in a clustering (unsupervised) operation, the question of labeling as a naming procedure cannot and should not be brought up directly. In general, clustering cannot provide naming. Only the external interpretation of the clusters can invest them with *meaning* as related to the problem. If the investigator wants to assign names to the clusters because he feels that the cluster results do reflect the state-of-nature classifications, he is free to do so; but the clustering algorithm itself cannot automatically do so with any guarantee that the results are meaningful.

As pointed out earlier, the use of the shared near neighbor similarity measure and pair-by-pair chaining is a clustering approach predominantly of the continuity type (for $k \ll n$) as differentiated from globular center of attraction types. In its pure form, it is not biased towards finding globular structures. In many applications, however, some degree of globular bias is not only allowable, but highly desirable, especially if the data is to be represented concisely.

It is interesting to consider the merging of continuity and globular approaches in clustering data; clearly, both approaches reflect certain attributes of the data. It may be valuable to mix the methods of approach so that, for a large range of problems, meaningful results have a high probability of occurrence. Alternatively, both approaches could be applied to the same set of data and the results compared.

One suggested approach to the mixing problem as related to the similarity measure proposed in this paper, and bearing in mind computational economy, is as follows:

Step 1: Use a first-level pair-by-pair chaining utilizing the neighborhood tables, but apply a restrictive threshold such that a large number of clumps result.

Step 2: Represent each clump by a point of mass equal to the number of points in each clump at the mean vector of the clump.

Step 3: Generate a new neighborhood table using the representative points as single neighbors but tabulating their mass along with the neighborhood table.

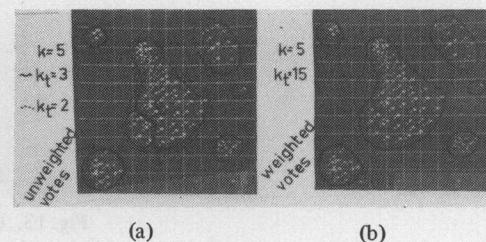


Fig. 11. Clustering results for example 1. (a) Example 1. (b) Example 1.

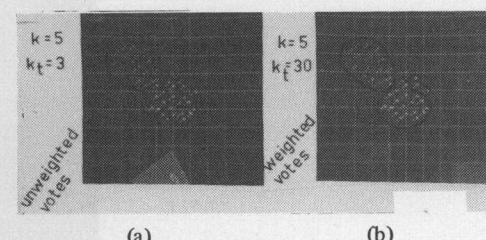


Fig. 12. Clustering results for example 2. (a) Example 2. (b) Example 2.

Step 4: Do a final clump-pair-by-clump-pair merge in which the masses of the neighbors are used in determining whether or not the corresponding pair should merge. Vote weighting by position could also be added.

Alternatively, simply apply the shared near neighbor rule pair by pair on the original data set but with a relatively large value of k . This increase in "neighborhood" size will tend to produce globular bias. Unfortunately, this neighborhood table requires much more storage than the method described above, in which the same core space could be used for both the original and the new neighborhood table if the number of points in the latter is less than half the original number of points. (Since a mass number is required for each new point.)

VII. EXAMPLES

In this section, a number of clustering results using the shared near neighbor algorithm (with and without vote weighting) on several data sets are presented. The first three examples were computed on an IBM 1130 and the data sets were generated manually via a joy stick and CRT display for two-dimensional vectors; the other examples were computed on a Data General Supernova.

In the first example [Fig. 11(a) and (b)], the data set contains two tight groups of sample points, one medium-tight group, one loose group, an isolated point, and a nonglobular group of points that are tightly spaced at the top with the spacing increasing towards the bottom. The autoscaling implied by the shared near neighbor rule is illustrated by the results of clustering this data set. The clustering results for $k = 5$ and k_t (threshold) = 3 and 2 are given in Fig. 11(a) for the unweighted vote case. The $k_t = 3$ result shows a fragmentation of the central group, while lowering the threshold to $k_t = 2$ keeps the central group as one cluster. Fig. 11(b) gives the clustering result for $k = 5$ and $k_t = 15$ where the multiplicative vote weighting of Section IV has been applied.

In the second example [Fig. 12(a) and (b)], a bridge exists between more or less globular groups of data points. Since

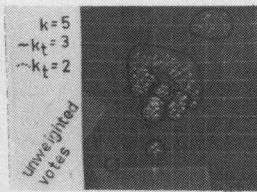
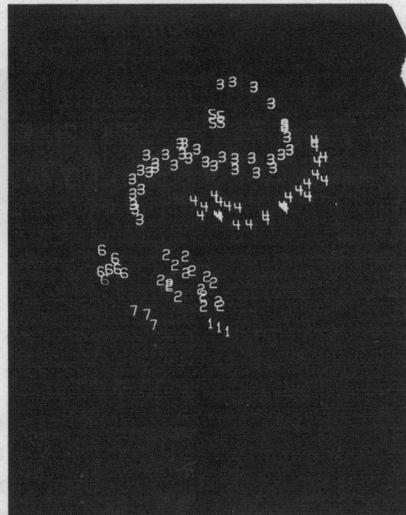
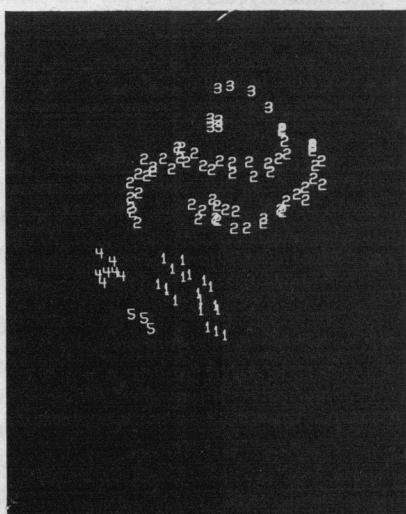


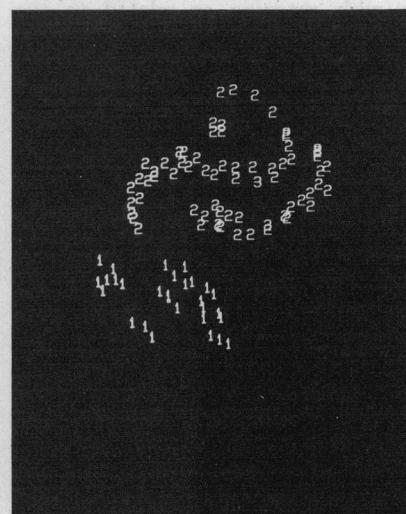
Fig. 13. Clustering results for example 3.



(a)



(b)



(c)

Fig. 14. Clustering results for example 4.

the shared near neighbor rule is not biased towards finding globular clusters, this is a difficult case for the method. Fig. 12(a) shows the poor result for $k = 5$, $k_t = 3$ with unweighted votes. However, if multiplicative vote weighting is used, the excellent results of Fig. 12(b) can be achieved ($k = 5$, $k_t = 30$). Close examination of the data set reveals that the third cluster indicated in Fig. 12(b) is not an unreasonable fragmentation.

The third example (Fig. 13) contains two isolated points, a tight group of three points, a loose group of five points, and a structure in the center that is subjectively difficult to resolve. The results using the shared near neighbor rule for $k = 5$ and $k_t = 3$ and 2 with unweighted votes is given in Fig. 13. The

cluster assignment seems reasonable enough on close inspection.

For the remaining examples clustering results are displayed with alphabetic labels to obviate grouping by boundary tracing which tends to obscure the evaluation.

The fourth example (Fig. 14) shows quite clearly the effects of variation of neighborhood size with respect to the continuity/globularity concept. Fig. 14(a) shows a fairly strict continuity basis clustering for $k = 5$ while Fig. 14(b) and (c) shows an increasing globular bias for neighborhood sizes of 10 and 15, respectively.

The fifth example shows an even more interesting result of

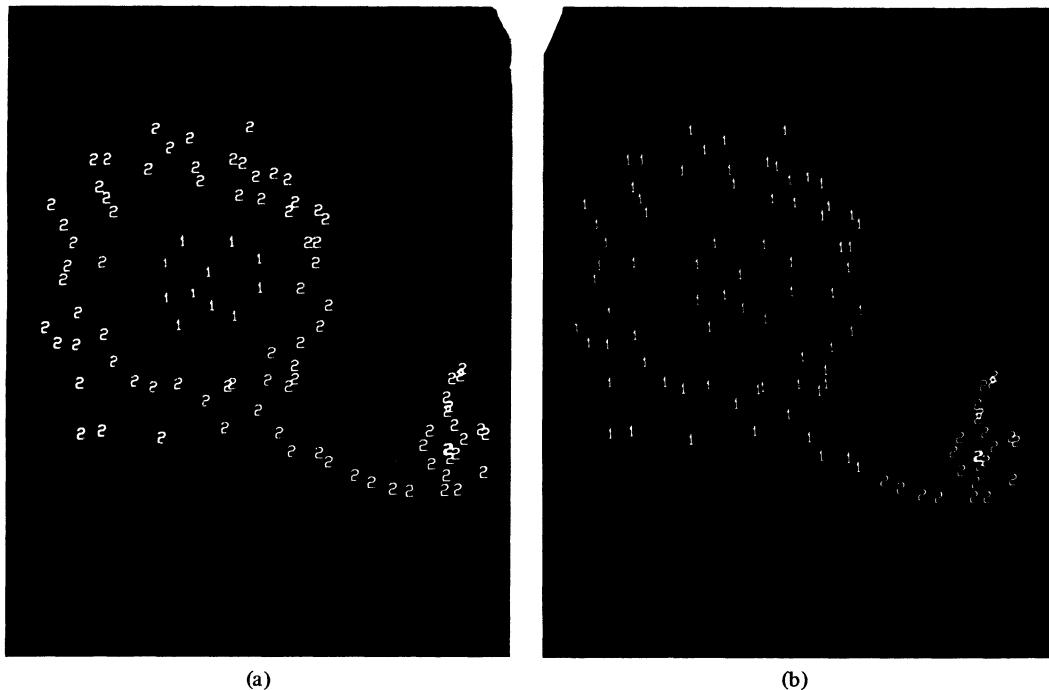


Fig. 15. Clustering results for example 5.

neighborhood size variation (Fig. 15). The more or less continuity-based clustering of Fig. 15(a) ($k = 7$) retains the tail in one cluster but separates a center group, while the result of Fig. 15(b) ($k = 15$) shows the two globular groups broken apart at the thin linking region.

The sixth example (Fig. 16) shows the ability of the shared nearest neighbor approach to solve the "two clusters touching at a neck" problem with a medium-sized neighborhood ($k = 10$). There is no distinct point pair distance inconsistency at the neck so that first-order chaining techniques would be unable to break the data at the neck point. Note also the point-to-point distance variations in the second cluster; the adaptive mechanism of the neighborhood approach is clearly evident.

The next example (Fig. 17) illustrates the cluster hierarchy approach to which the shared near neighbor approach is particularly amenable. In Fig. 17(a)-(d), a neighborhood size of 20 is used. In (a) ($k_t = 18$) subclusters appear; in (b) ($k_t = 15$) two of the subclusters of (a) have merged; in (c) ($k_t = 12$) further merging has taken place; in (d) ($k_t = 8$) superclusters are now evident.

The final example (Fig. 18) shows a link between two globular groups breaking for a medium-sized neighborhood ($k = 10$) for a threshold change from $k_t = 5$ to $k_t = 7$.

The computational times for neighborhood table evaluation and clustering calculations are given for the examples of Figs. 14-18 for a Data General Supernova minicomputer with hardware integer multiply/divide but no floating point hardware (Fig. 19). There are a number of points worth noting when examining these times. Firstly, since $L = 2$ in all cases presented, the neighborhood table calculation time is much smaller than for a more realistic dimension of the data (e.g., $L = 10$ to 50); consequently, the feature of the method that allows a single neighborhood calculation for multiple

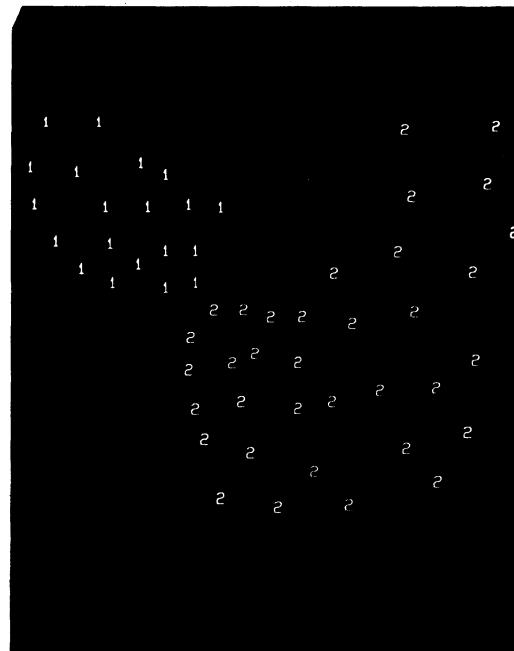


Fig. 16. Clustering results for example 6.

clustering passes is not shown to the same advantage as it would for large L . Secondly, the neighborhood table would normally be evaluated for the largest k value of interest and part of it would not be used for lower k explorations. Finally the $(k + 1)^2$ factor in the clustering time analysis is obscured by the link merge time and the short cuts that can be taken whenever, for a pair of candidates, each cannot be found in the other's near neighbor list.

These examples give some indication of the behavior of the shared near neighbor rule for a variety of types of data; its usefulness is restricted to those applications where the same

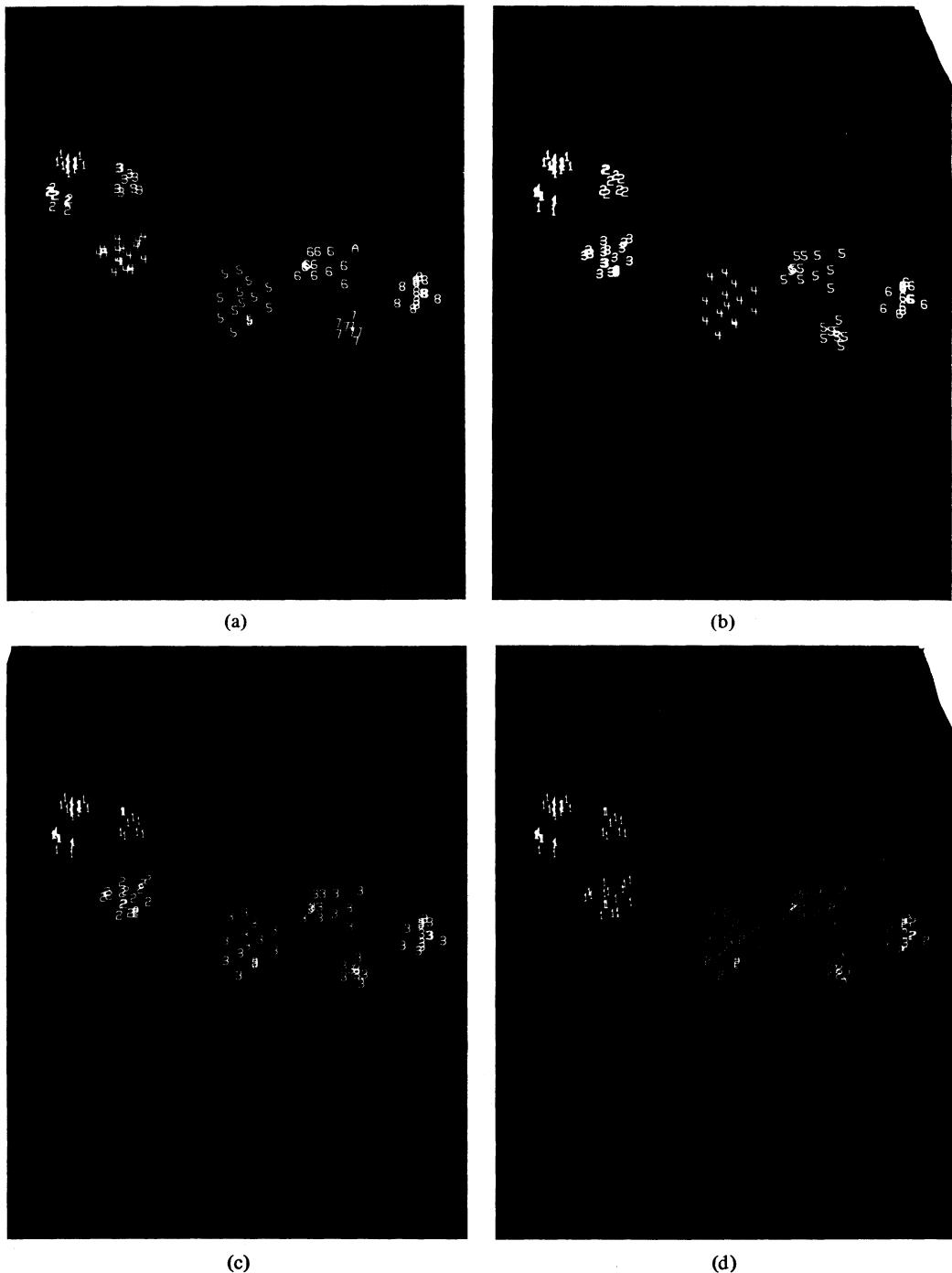


Fig. 17. Clustering results for example 7.

type of results as in the above examples are acceptable to the investigator.

VIII. CONCLUSIONS

A nonparametric clustering algorithm that uses the concept of similarity based on the sharing of near neighbors has been described; several variations of method of application are discussed, and a number of examples to illustrate the potentials of the rule presented. The rule is extremely simple to apply and is economic with regard to both processor time and storage. The rule is applicable to a wide class of practical

problems involving large sample size and high dimensionality, and is particularly recommended as a first analysis tool when little *a priori* knowledge of the structure of the data is known. As more is learned of the physical process that generated the data, more sophisticated methods reflecting this knowledge should be preferred; nevertheless this method should be among the clustering tools available for data analysis and be used both for data structure exploration and for data compression.

It is interesting to compare our approach with a method developed by Zahn [15], which starts with the development of the minimal spanning tree (MST) and follows with various techniques of finding "inconsistent" edges that can be removed

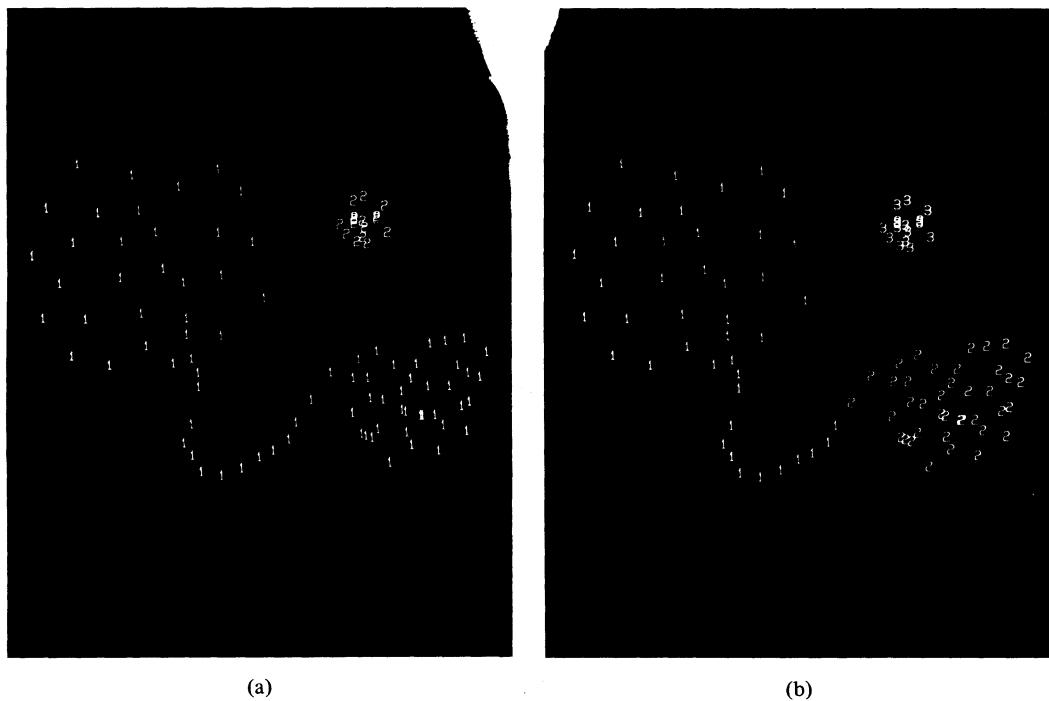


Fig. 18. Clustering results for example 8.

Figure	n	k	k_t	T_1 (Secs)	T_2 (Secs)
14(a)	99	5	2	35	8
14(b)	99	10	7	47	20
14(c)	99	20	15	83	79
15(a)	99	7	4	40	12
15(b)	99	15	11	62	45
16	55	10	7	19	8
17(a)	99	20	18	79	79
17(b)	99	20	15	79	70
17(c)	99	20	12	79	62
17(d)	99	20	8	79	51
18(a)	99	10	5	48	19
18(b)	99	10	7	48	20

Fig. 19. Table of computing times for examples. $L = 2$ in all cases. T_1 : computational time for neighborhood table evaluation; T_2 : clustering and link merge calculation time.

to leave disjoint subgraphs that represent the cluster groups. Although the concept of "nearest neighbor" is inherent in the development of the minimal spanning tree, there is nothing in the method to suggest the use of shared near neighbors as a measure of similarity. Zahn's approach is to discover dissimilarity of subgraphs through the recognition of relatively large spanning branches, neck points, and discontinuities with respect to the minimal spanning tree's coverage of the data points. Zahn's method is particularly suited to the clustering of data with linear chains and elongated cluster shapes but would perhaps be less versatile than the shared nearest neighbor approach for cases where strict continuity is not the

only criterion of cluster cohesion or where certain types of "*a priori*" cluster shape hypotheses are required to be tested against the data. Both methods do, however, share the conviction that adaptive distance scaling is of importance in clustering.

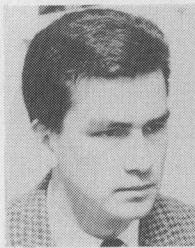
The computational expense of finding Zahn's minimal spanning tree is considerably greater than that of constructing the near neighbor table. There are $n(n - 1)/2$ edges to the graph; where n is large, storage of these edge distances is prohibitive and one must resort to sequential procedures (Zahn's [15, algorithm 2]), which consist of $\sum_{I=0}^n [(n - I) \times I]$ distance evaluations and comparisons. The computation complexity is of the order of $\{(n - 1)(n)(n + 1)\}/(6)\} L$, which is a third power of n evaluation. The computational complexity of setting up the near neighbor table is approximately of the order of $n^2 L$, so that the complexity ratio for finding the minimal spanning tree over that of constructing the near neighbor table is $\approx n/6$, which cannot be overlooked for large n .

Zahn's method is based on the manner in which the MST represents structure in the data; as the MST is essentially chain structured with simple point pair distances governing that structure, the method is more or less of the "first order" as globular multipoint groupings are not represented by the MST in a meaningful way. The shared near neighbors approach allows for variation of point group sensitivity inherent in the concept of neighborhood and votes, an adaptive electorate concept with range control. The voting groups can be extremely localized to emphasize continuity or be somewhat global to emphasize centers of mass and the globular concept of cluster definition. This local/global voting group variation can be explored inexpensively on a common neighborhood table that is calculated only once.

Some areas for further investigation are: 1) detailed speed and storage comparison with other clustering methods; 2) the theoretical analysis of the rule as related to Bayes, k th nearest neighbor rules, and empirical density estimates; and 3) the investigation of means for conveniently inserting *a priori* problem knowledge when it is available; related to 2) and 3) is 4) the investigation of various vote weighting functions suggested in Section IV which are beyond the scope of this paper.

REFERENCES

- [1] E. A. Patrick and F. P. Fischer, "Cluster mapping with experimental computer graphics," *IEEE Trans. Comput.*, vol. C-18, pp. 987-991, Nov. 1969.
- [2] E. A. Patrick and L. Shen, "Interactive use of problem knowledge for clustering and decision making," *IEEE Trans. Comput.*, vol. C-20, pp. 216-222, Feb. 1971.
- [3] J. M. Sammon, Jr., "Interactive pattern analysis and classification," *IEEE Trans. Comput.*, vol. C-19, pp. 594-616, July 1970.
- [4] E. A. Patrick, "On a class of unsupervised estimation problems," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 407-415, May 1968.
- [5] G. H. Ball, "Data analysis in the social sciences: What about the details?" in *1965 Fall Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 27. Washington, D.C.: Spartan, 1965, pp. 533-559.
- [6] G. H. Ball and D. J. Hall, "ISODATA, An iterative method of multivariate analysis and pattern classification," presented at the Int. Commun. Conf., Philadelphia, Pa., 1966.
- [7] R. N. Shepard and J. O. Carroll, "Parametric representation of nonlinear data structures," in *Multivariate Analysis*, P. R. Krishnaiah, Ed. New York: Academic, 1966.
- [8] A. A. Dorofeyuk, "Teaching algorithm for a pattern recognition machine without a teacher, based on the method of potential functions," *Automat. Remote Contr. (USSR)*, vol. 27, pp. 1728-1737, Oct. 1966.
- [9] K. Fukunaga and W. L. G. Koontz, "A criterion and algorithm for grouping data," *IEEE Trans. Comput.*, vol. C-19, pp. 917-923, Oct. 1970.
- [10] I. Gitman and M. D. Levine, "An algorithm for detecting unimodal fuzzy sets and its application as a clustering technique," *IEEE Trans. Comput.*, vol. C-19, pp. 583-593, July 1970.
- [11] I. Gitman, "A parameter free clustering model," *Pattern Recognition*, vol. 4, pp. 307-315, 1972.
- [12] G. Sebestyen, *Decision Making Processes in Pattern Recognition* (ACM Monograph). New York: Macmillan, 1962.
- [13] R. E. Bonner, "On some clustering techniques," *IBM J. Res. Develop.*, vol. 8, pp. 22-32, Jan. 1964.
- [14] R. R. Sokal and P. H. A. Sneath, *Principles of Numerical Taxonomy*. San Francisco, Calif.: Freeman, 1963.
- [15] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. C-20, pp. 68-86, Jan. 1971.



R. A. Jarvis was born in Rangoon, Burma, on January 9, 1941. He received the B.S. degree in electrical engineering and the Ph.D. degree, both from the University of Western Australia, Nedlands, in 1962 and 1968, respectively.

During 1966-1967 he was Senior Tutor, and in 1968 Temporary Lecturer, in the Department of Electrical Engineering, University of Western Australia. In 1968-1969 he was an Assistant Professor at Purdue University, Lafayette, Ind. He is currently Senior Lecturer in Computer

Science at the Australian National University, Canberra. His areas of interest include automata theory, logic design, adaptive control, and pattern recognition.



Edward A. Patrick (M'65-SM'73) was born in Wheeling, W. Va., in 1937. He received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1960 and 1962, respectively, and the Ph.D. degree in electrical engineering from Purdue University, Lafayette, Ind., in 1966. Beginning in 1970 he took multiple leaves including a Sabbatical from Purdue University to obtain the M.D. degree.

From 1958 to 1961 he was with the M.I.T. Instrumentation Laboratory; from 1961 to 1963 he was with Sylvania Applied Research Laboratory; and the summer of 1964 he was with Melpar Corporation. He has been a consultant to Texas Instrument Corporation, Magnavox Corporation, DuPont Corporation, the Department of the Navy, the Department of the Air Force, Technology Service Corporation, the Department of Health, Education, and Welfare, and the Regenstrief Institute for Health Care Delivery. In 1966 he became an Assistant Professor of Electrical Engineering at Purdue University and rose to Associate Professor in 1968. He is the author of *Fundamentals of Pattern Recognition* (Prentice-Hall, 1972) and of over 60 technical papers, and is a contributor to *Frontiers of Pattern Recognition*.

Dr. Patrick is a member of the American Society for Engineering Education, the American Medical Association, Sigma Xi, and Eta Kappa Nu. He is a member of the Ad Com Committee, the IEEE Systems, Man, and Cybernetics Society, Chairman of that Society's Pattern Recognition Committee, Past Chairman of the Chicago Joint Sections Information Theory Group, and on the Program Committee for the first International Conference on Pattern Recognition. He has been chairman of several IEEE organized workshops on pattern recognition and is an Associate Editor of *Computers in Biology and Medicine*.