UNSUPERVISED MULTIDIMENSIONAL HIERARCHICAL CLUSTERING

Rakesh Dugad and Narendra Ahuja

Department of Electrical and Computer Engineering
Beckman Institute, University of Illinois, Urbana, IL 61801.
dugad@uiuc.edu

ABSTRACT

A method for multidimensional hierarchical clustering that is invariant to monotonic transformations of the distance metric is presented. The method derives a tree of clusters organized according to the homogeneity of intracluster and interpoint distances. Higher levels correspond to coarser clusters. At any level the method can detect clusters of different densities, shapes and sizes. The number of clusters and the parameters for clustering are determined automatically and adaptively for a given data set which makes it unsupervised and non-parametric. The method is simple, noniterative and requires low computation. Results on various sample data sets are presented.

1. INTRODUCTION

The problem of hierarchical clustering in multidimensional sample space finds applications in many areas. This paper presents an unsupervised and non-parametric method for hierarchical clustering in high dimensional data sets. There are many methods which give clusters for chosen parameters and others which do provide a hierarchy of clusters. Some of the most popular algorithms available for clustering are k-means algorithm, graph-theoretical algorithms and linkage methods [1].

The k-means algorithm starts with an arbitrary partition and consists of alternately computing the centroids of the partitions at each iteration and reassigning the samples so as to minimize the sum-total of variances of individual partitions. Because of its dependence on the centroid this method is more suitable for detecting compact and globular clusters. For example it can not detect two distinct clusters having approximately the same centroid like those shown in figure 1(e). Also it is not invariant to monotonic transformations of the proximity matrix. This means that one has to be careful in scaling and combining different variables into a proximity measure[2].

Zahn [3] gives a number of graph-theoretical algorithms based on MST (minimum spanning tree). The algorithm consists in finding the MST of the given pattern , identifying and deleting inconsistent edges in the MST and forming connected components of edges to get the clusters. The method works good on various data sets including non-spherical clusters and clusters with smoothly varying point densities but special heuristics are needed to detect inconsistent edges in complex situations e.g. in the case of two homogeneous clusters of slightly different point densities shown in figure 1(a) the sparse cluster will have many inconsistent edges. Zahn suggests detecting and deleting the denser cluster first and then clustering the remaining data. Also prior knowledge of the shapes of the clusters is needed to get the proper heuristic for identifying the inconsistent edges[1]. This can be a problem in more than two dimensions.

Methods based on Voronoi neighborhood have also been proposed [4, 5] and shown to yield good results for various kinds of clusters. However such methods have been developed mainly for two-dimensional data due to the computational difficulties in higher dimensions[1].

The above methods are inherently partitional. The single and complete-linkage methods[2] and Ward's method [6] are some commonly used agglomerative hierarchical clustering algorithms. In the single-linkage method the similarity between two clusters is judged by the most similar sample points, one in each cluster, whereas in the complete-linkage method it is judged by the least similar points. In Ward's method clusters whose union results in minimum loss of 'information' are combined at each step. Though these methods have several desirable theoretical properties they are biased toward finding spherical clusters even though the data contains clusters of other shapes. Also Ward's method performs better when the clusters are approximately same size than when they are of different sizes. Single and complete linkage methods are invariant to monotonic transformations of the proximity matrix but Ward's method is not[2].

In this paper we present a method for hierarchical clustering aimed at removing some of the difficulties mentioned above. The method can detect spherical or non-spherical clusters and these clusters do not have to be compact, i.e., there can be holes in them as shown in figure 1(e). Also the notion of density is built in the method and hence it can robustly detect even homogeneous clusters that are close by but differ in point density. The method is unsupervised and non-parametric- it does not need to know the number of clusters *a priori*. It is applicable to multi-dimensional data as well. The method is invariant under monotonic transformations of the proximity matrix because it depends only on the relative ranks of the sample points. Moreover the method is simple, deterministic (which implies that it does not depend on the order in which the points are scrutinized) and has low computational costs.

2. OVERVIEW AND MOTIVATION OF THE ALGORITHM

The algorithm consists of deciding on the neighbors of each point and then finding connected components to get the clusters in the given data. The data to be clustered is a set of NL-dimensional points on which a suitable metric d is defined. The input to the algorithm is the proximity matrix (i.e. the d value for all pairs of points) of this set and the output is a hierarchy of clusters. Our definition of

The support of the office of Naval Research under grant N00014-96-1-0502 is greatfully acknowledged

the neighborhood of a given point depends on what is called as the mutual neighborhood value (mnv) [7] between two given points. Let P and Q be two points of the given data set. If P is the mth nearest neighbor of Q and Q is the nth nearest neighbor of P then the mnv between P and Q is defined to be $m + n$. The main motivation for this definition is that two points P and Q have a higher tendency to group if not only P is close to Q but also Q is close to P. The mnv is a semi-metric and does not satisfy the triangle inequality.

We first present a basic scheme and then motivate two important improvements to this scheme which lead to our final algorithm. A parameter M_T is used to denote the acceptability of points as neighbors in terms of their mnv. Given a point P belonging to the data set all points Q such that mnv$(P, Q) \leq M_T$ are said to belong to the neighborhood of P. This gives us a neighborhood graph on the data set whose connected components are the required clusters. This is similar to the scheme described in [7] but there are two modifications that make it attractive as explained below.

First consider figure 1(a) which shows two clusters of different densities that are close by. Points P and Q have an mnv of 7 between them. So if M_T is say 8 (which is not very large) then Q will be called neighbor of P (and vice-versa) and this one link would merge the two clusters. Point K has only an mnv of 6 w.r.t. Q even though it is at a larger distance from Q than P is. This happens because P has a higher density of points around it and hence is likely to belong to a different cluster. This suggests that we should stipulate that if mnv$(Q, P) >$ mnv(Q, K) then $d(Q, P) > d(Q, K)$. Else P should be regarded as not belonging to the neighborhood of Q. We shall call such a point P invalid w.r.t. Q .

Second point to note is the dual of the point mentioned above. Q will never be marked invalid w.r.t. P because of the large distance of Q from P (compared to other points close to P) and hence its larger mnv w.r.t. P seems justified. Hence under our basic scheme with just the above modification Q will belong to the neighborhood of P and hence the two clusters will merge. Hence we need to stipulate that a point Q will not belong to the neighborhood of point P if P does not belong to the neighborhood of Q (i.e. if P is invalid w.r.t. Q).

This explains the two modifications we make. Then the algorithm just consists of creating a neighborhood graph on the given data (keeping in mind the above two stipulations and the threshold M_T on mnv) and finding connected components. Note how the above two stipulations help detect clusters of different point densities more robustly than simply using the mnv. Also the method is invariant to monotonic transformations of the proximity matrix since the mnv and the above two stipulations depend only on the relative distances among the points (since if P is at a larger (or smaller) distance from Q it will also remain so under any monotonic transformation). Further since we only use the metric d the method is equally applicable in more than two dimensions.

The above observations are formalized in the definition of neighborhood of a given point: a point P belongs to the neighborhood of point Q if and only if *all* the following conditions hold :

1. mnv$(Q, P) \leq M_T$;
2. There exists no point K s.t. mnv$(Q, K) <$mnv(Q, P) but $d(Q, K) \geq d(Q, P)$. A point P violating this constraint is called *invalid* w.r.t. Q.
3. Q is not be invalid w.r.t. P.

It should be noted that the last two stipulations can give rise to some small clusters because of the strict condition that neither point should regard the other as invalid if they are to be neighbors of each other. But at the same time these stipulations also make it possible to identify problematic clusters like those that are close by but have slightly different densities. To address this we do some post processing of the small clusters (say of size ≤ 5) : we check to which clusters individual points of the small cluster would have belonged if *one* of the last two stipulations is relaxed. The smaller cluster is merged with larger cluster which gets maximum number of votes. Note that if a small cluster is perceptually distinct from other clusters than *all* of its points would satisfy the last two stipulations and hence this cluster would not be merged with any larger cluster. Also note that we do not just activate all the links that come up when one of the last two constraints is relaxed because this can cause two bigger clusters to each merge with the smaller one and thus merge themselves in one cluster.

3. THE HIERARCHY

The final clusters we get depend on the parameter M_T . If M_T is increased the clusters start merging. Note that increasing M_T can only merge clusters but can never split them. Therefore if for a certain range of M_T values the number of clusters remains the unchanged then the *actual* clusters corresponding to these values of M_T must be the same. This suggests that if we plot a graph of the number of clusters vs. the M_T value at which those many clusters are obtained (we shall call it the *stability curve*) then any constant levels (plateaus) of this graph will indicate that clusters have not changed even though the M_T is increased. Therefore such clusters denote a valid partition of the data at some scale. These clusters form a level of the cluster hierarchy indexed by the corresponding range of M_T values. Parts of the stability curve containing no plateaus are *transients* where some clusters merge as M_T is increased. Such clusters and the corresponding M_T values are discarded – they do not correspond to valid groupings at any scale.

4. THE ALGORITHM

The algorithm consists of the following steps:

Step 1 (Sorting the distance pairs) : Let $D = [d(i, j)]$ denote the proximity matrix and let $m = M_T - 1$. Since M_T is the maximum allowd value of mnv between two neighbors, we need to consider only upto mth nearest neighbors of a given point. Sort D to identify m nearest neighbors. Let the matrix thus got be denoted by $R = [r_{ij}], i = 0$ to $N - 1, j = 0$ to $m - 1$ where r_{ij} is the jth nearest neighbor of i.

Step 2 (Finding the mnvs) : R contains entries for a maximum of Nm pairs of points. Use R to construct a matrix containing mnvs of these pairs of points.

Step 3 (Sorting the mnvs) : Sort the mnvs computed in step 2. Let the matrix containing the sorted mnvs be called $V = [v_{ij}], i = 0$ to $N - 1, j = 0$ to $m - 1$ where v_{ij} is the point that is jth closest to i in terms of mnv of i with all points.

Step 4 (Making neighbors) : Let $i = 0, j = 1$ and $r = 0$. If mnv$(i, v_{ij}) \leq M_T$ and $d(i, v_{ij}) > r$ then mark v_{ij} as the neighbor of i and $r \leftarrow d(i, v_{ij})$. $j \leftarrow j + 1$. If $j >= m$ or mnv$(i, v_{ij}) > M_T$ then $i \leftarrow i + 1$. If $i >= N$ stop else repeat this step.

Note that we have taken care of conditions 1 and 2. Condition 3 is now enforced by removing any neighborhood relations that are not symmetric.

Step 5 (Finding connected components) : Find connected components on the neighborhood graph obtained above and identify any small clusters.

Step 6 (Post processing) : If there are any small clusters then find out if they would merge with any larger cluster if one of conditions 2 and 3 is relaxed as described in section 2 and modify the neighborhood links accordingly. Hence for each point of the small clusters we need to find out which all of its m nearest neighbors would belong to its neighborhood if the condition mentioned is relaxed. Connected components of this graph give the required clusters.

5. COMPUTATIONAL COMPLEXITY

In step 1 for a given point it takes $O(N \log m)$ to get the m nearest neighbors and then $O(m \log m)$ to arrange them in ascending order of their distances. Hence this step has a total complexity of $O(((N + m) \log m)N)$ for all N points. In step 2 for each entry in R matrix we have to search in the worst case a list of m points in the other row to find out the mnv. Hence the worst case complexity of this step case is $O(Nm^2)$. Step 3 has $O(N \times m \log m)$ complexity. In step 4 we need to visit m points in worst case around any point to get its neighbors. Hence this step has $O(Nm)$ complexity. Steps 5 and 6 each have $O(Nm)$ complexity.

Note that m will be usually much smaller compared to N and hence the major computational load is in the first part in sorting the distances. This computation may be reduced if some organization of the data already exists for the particular application. Also note that for computing the stability curve we need to repeat only the last three steps which are $O(Nm)$ complexity.

6. RESULTS

Figure 1(a)(left) curve for the data points shown in figure 1(a)(right). The stability curve is shown only starting from an M_T of 5 because there are no plateaus before that. Let n denote number of clusters. We see that we have a distinct plateau at $n = 2$. The corresponding clusters are shown in figure 1(a)(right). We also conducted experiments to see how the stability curve would look like if the clustering was done only on the basis of mnv (i.e. using only condition 1). It was found that in that case the stability curve had no plateaus except for $n = 1$. This illustrates the crucial role played by the other two conditions in deciphering valid clusters.

Similar results for another data set are shown in figure 1(b). Here we have plateaus for $n = 3$ and 2. The results for $n = 3$ are shown in figure 1(b). At the next plateau corresponding to $n = 2$ the clusters shown by o and + merge. We can argue that this is desired because the density of o points is closer to that of + points than * points. The algorithm achieves this because the concept of invalid points (in conditions 2 and 3) captures the notion of density.

Figure 1(c) shows the results for another data set. Here we have a large number of plateaus in the stability curve. The plateau at $n = 15$ corresponds the 15 perceptually distinct clusters in the data set which can be easily perceived in the figure. This corresponds to the finest level of detail. Figure 1(c) shows the grouping corresponding to the plateau at $n = 2$. We can note that this is in fact what one would perceive when seen from a very course level.

Finally figures 1(d) & (e) shows the results for two more data set. The results for the $n = 2$ plateaus (which are the only significant plateaus) are shown in figures 1(d) and 1(e). Again we see that the clusters have been captured successfully.

7. CONCLUSION

An unsupervised multidimensional hierarchical clustering algorithm that is able to detect clusters with different densities in a more robust way compared to using only the mutual neighborhood value is developed. Results on various data sets have been presented which illustrate the efficacy of the method in detecting clusters of various kinds. Although results for only 2-D are presented the method is applicable just as well in higher dimensions. The salient features of the algorithm are:

1. It can robustly detect clusters that are close by and differ in point density. It has no difficulty in detecting clusters that are well separated.
2. It gives a hierarchy of clusters that are perceptually meaningful.
3. It is invariant to monotonic transformations of the distance metric. Hence the difficulties involved with scaling and combining different variables into the distance metric are of less concern.
4. It is not limited spherical or equal sized clusters or any particular metric.
5. It does not require any user specified parameters such as the expected number of clusters or a starting classification as required by many other algorithms.
6. It does not depend on the order in which the points are processed.
7. It is simple, non-iterative and has low computation cost.

8. REFERENCES

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, New Jersey 07632: Prentice Hall, 1988.

[2] B. S. Everitt, *Cluster Analysis*. Halsted Press, third ed., 1993.

[3] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers*, vol. C-20, pp. 68–86, January 1971.

[4] N. Ahuja, "Dot pattern processing using voronoi neighborhoods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, pp. 336–343, 1982.

[5] N. Ahuja and M. Tuceryan, "Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component gestalt," *Computer Vision, Graphics, and Image Processing*, vol. 48, no. 3, pp. 304–356, 1989.

[6] J. H. Ward, "Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58, pp. 236–244, 1963.

[7] K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighborhood," *Pattern Recognition*, vol. 10, pp. 105–112, 1978.

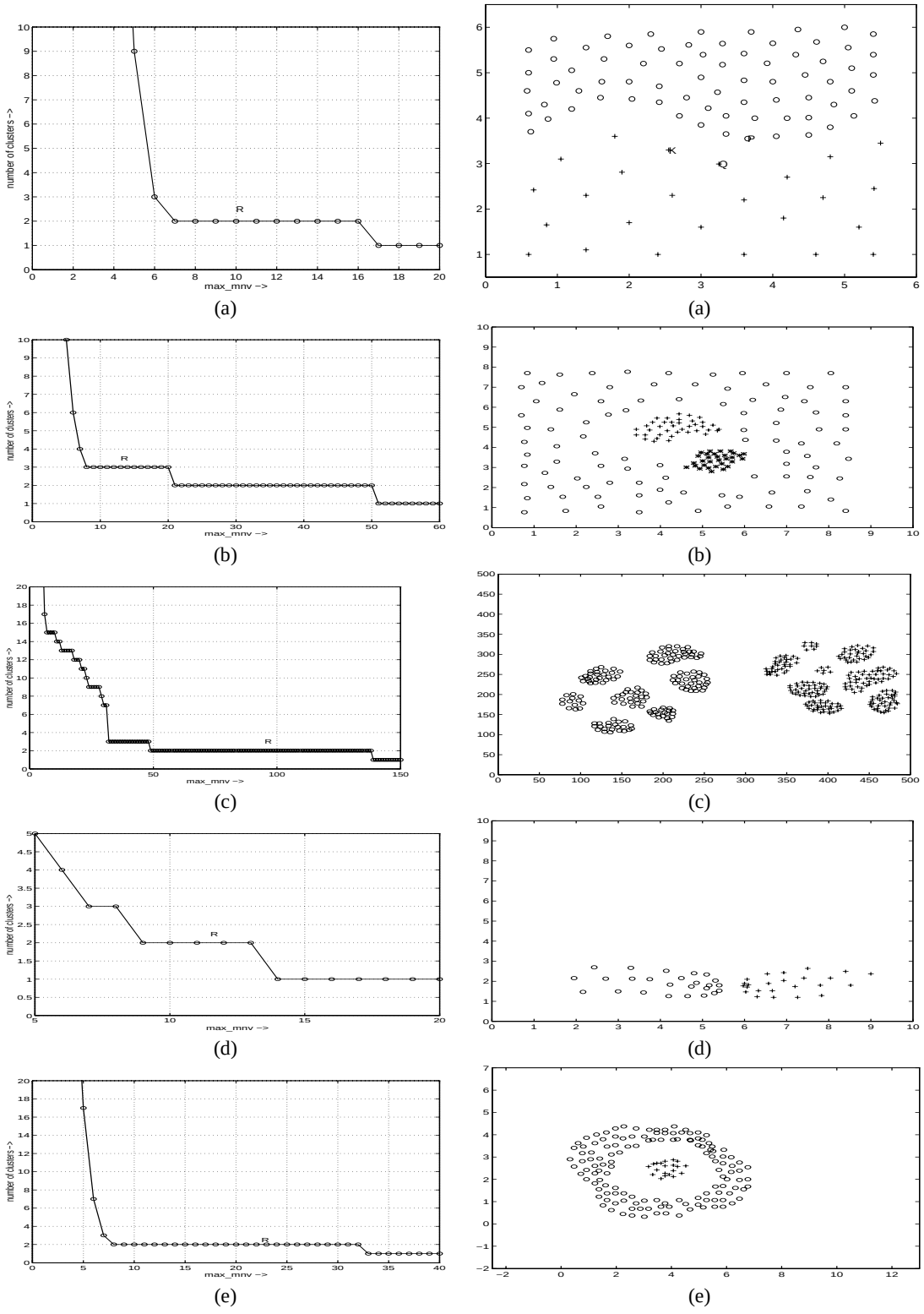

Figure 1: Left figures show the stability curve and the right figures show the clusters corresponding to the plateau marked R on the stability curve.