

# The Challenges of Clustering High Dimensional Data<sup>\*</sup>

Michael Steinbach, Levent Ertöz, and Vipin Kumar

## Abstract

Cluster analysis divides data into groups (clusters) for the purposes of summarization or improved understanding. For example, cluster analysis has been used to group related documents for browsing, to find genes and proteins that have similar functionality, or as a means of data compression. While clustering has a long history and a large number of clustering techniques have been developed in statistics, pattern recognition, data mining, and other fields, significant challenges still remain. In this chapter we provide a short introduction to cluster analysis, and then focus on the challenge of clustering high dimensional data. We present a brief overview of several recent techniques, including a more detailed description of recent work of our own which uses a concept-based clustering approach.

## 1 Introduction

Cluster analysis [JD88, KR90] divides data into meaningful or useful groups (clusters). If meaningful clusters are the goal, then the resulting clusters should capture the “natural” structure of the data. For example, cluster analysis has been used to group related documents for browsing, to find genes and proteins that have similar functionality, and to provide a grouping of spatial locations prone to earthquakes. However, in other cases, cluster analysis is only a useful starting point for other purposes, e.g., data compression or efficiently finding the nearest neighbors of points. Whether for understanding or utility, cluster analysis has long been used in a wide variety of fields: psychology and other social sciences, biology, statistics, pattern recognition, information retrieval, machine learning, and data mining.

In this chapter we provide a short introduction to cluster analysis, and then focus on the challenge of clustering high dimensional data. We present a brief overview of several recent techniques, including a more detailed description of recent work of our own which uses a concept-based approach. In all cases, the approaches to clustering high dimensional data must deal with the “curse of dimensionality” [Bel61], which, in general terms, is the widely observed phenomenon that data analysis techniques (including clustering), which work well at lower dimensions, often perform poorly as the dimensionality of the analyzed data increases.

## 2 Basic Concepts and Techniques of Cluster Analysis

### 2.1 What Cluster Analysis Is

Cluster analysis groups objects (observations, events) based on the information found in the data describing the objects or their relationships. The goal is that the objects in a

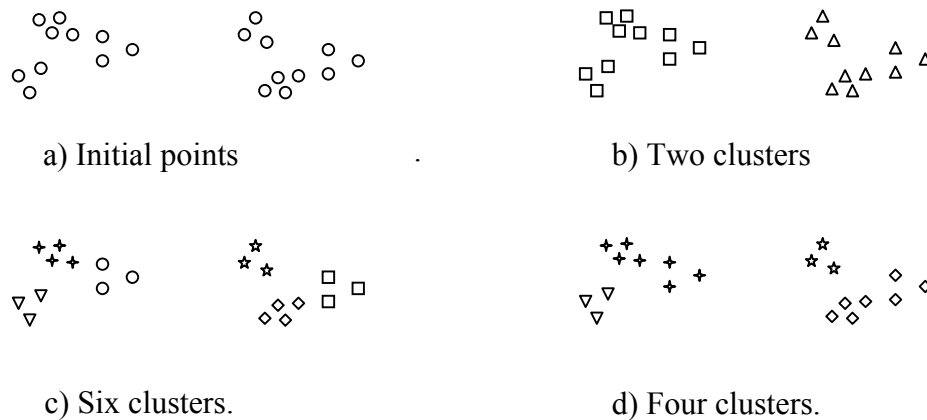
---

<sup>\*</sup> This research work was supported in part by the Army High Performance Computing Research Center cooperative agreement number DAAD19-01-2-0014. The content of this paper does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Access to computing facilities was provided by AHPARC and the Minnesota Supercomputing Institute.

group should be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better the clustering.

The definition of what constitutes a cluster is not well defined, and in many applications, clusters are not well separated from one another. Nonetheless, most cluster analysis seeks, as a result, a crisp classification of the data into non-overlapping groups. Fuzzy clustering [HKKR99] is an exception to this, and allows an object to partially belong to several groups.

To illustrate the difficulty of deciding what constitutes a cluster, consider Figure 1, which shows twenty points and three different ways that these points can be divided into clusters. If we allow clusters to be nested, then the most reasonable interpretation of the structure of these points is that there are two clusters, each of which has three subclusters. However, the apparent division of the two larger clusters into three subclusters may simply be an artifact of the human visual system. Finally, it may not be unreasonable to say that the points form four clusters. Thus, we again stress that the notion of a cluster is imprecise, and the best definition depends on the type of data and the desired results.



**Figure 1:** Different clusterings for a set of points.

## 2.2 What Cluster Analysis Is Not

Cluster analysis is a classification of objects from the data, where by “classification” we mean a labeling of objects with class (group) labels. As such, clustering does not use previously assigned class labels, except perhaps for verification of how well the clustering worked. Thus, cluster analysis is sometimes referred to as “unsupervised classification” and is distinct from “supervised classification,” or more commonly just “classification,” which seeks to find rules for classifying objects given a set of pre-classified objects. Classification is an important part of data mining, pattern recognition, machine learning, and statistics (discriminant analysis and decision analysis).

While cluster analysis can be very useful, either directly or as a preliminary means of finding classes, there is more to data analysis than cluster analysis. For example, the decision of what features to use when representing objects is a key activity of fields such as data mining, statistics, and pattern recognition. Cluster analysis typically takes the features as given and proceeds from there. Thus, cluster analysis,

while a useful tool in many areas, is normally only part of a solution to a larger problem that typically involves other steps and techniques.

## 2.3 The Data Matrix

Objects (measurements, events) are usually represented as points (vectors) in a multi-dimensional space, where each dimension represents a distinct attribute (variable, measurement) describing the object. For simplicity, it is usually assumed that values are present for all attributes. (Techniques for dealing with missing values are described in [JD88, KR90].) Thus, a set of objects is represented (at least conceptually) as an  $m$  by  $n$  matrix, where there are  $m$  rows, one for each object, and  $n$  columns, one for each attribute. This matrix has different names, e.g., pattern matrix or data matrix, depending on the particular field.

The data is sometimes transformed before being used. One reason for this is that different attributes may be measured on different scales, e.g., centimeters and kilograms. In cases where the range of values differs widely from attribute to attribute, these differing attribute scales can dominate the results of the cluster analysis, and it is common to standardize the data so that all attributes are on the same scale. A simple approach to such standardization is, for each attribute, to subtract of the mean of the attribute values and divide by the standard deviation of the values. While this is often sufficient, more statistically “robust” approaches are available, as described in [KR90].

Another reason for initially transforming the data is to reduce the number of dimensions, particularly if the initial number of dimensions is large. We defer this discussion until later in this chapter.

## 2.4 The Proximity Matrix

While cluster analysis sometimes uses the original data matrix, many clustering algorithms use a similarity matrix,  $S$ , or a dissimilarity matrix,  $D$ . For convenience, both matrices are commonly referred to as a proximity matrix,  $P$ . A proximity matrix,  $P$ , is an  $m$  by  $m$  matrix containing all the pairwise dissimilarities or similarities between the objects being considered. If  $x_i$  and  $x_j$  are the  $i^{\text{th}}$  and  $j^{\text{th}}$  objects, respectively, then the entry at the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of the proximity matrix is the similarity,  $s_{ij}$ , or the dissimilarity,  $d_{ij}$ , between  $x_i$  and  $x_j$ . For simplicity, we will use  $p_{ij}$  to represent either  $s_{ij}$  or  $d_{ij}$ . Figures 2a, 2b, and 2c show, respectively, four points and the corresponding data and proximity (distance) matrices. (Different types of proximities are described in Section 2.7.)

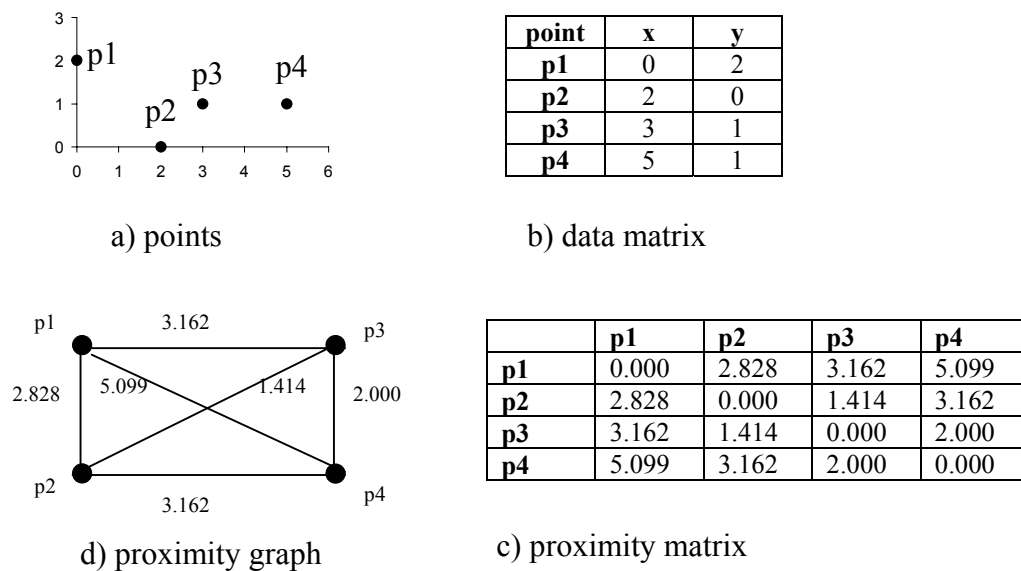
For completeness, we mention that objects are sometimes represented by more complicated data structures than vectors of attributes, e.g., character strings or graphs. Determining the similarity (or differences) of two objects in such a situation is more complicated, but if a reasonable similarity (dissimilarity) measure exists, then a clustering analysis can still be performed. In particular, clustering techniques that use a proximity matrix are unaffected by the lack of a data matrix.

## 2.5 The Proximity Graph

A proximity matrix defines a weighted graph, where the nodes are the points being clustered, and the weighted edges represent the proximities between points, i.e., the entries of the proximity matrix (see Figure 2c). While this proximity graph can be

directed, which corresponds to an asymmetric proximity matrix, most clustering methods assume an undirected graph. Relaxing the symmetry requirement can be useful in some instances, but we will assume undirected proximity graphs (symmetric proximity matrices) in our discussions.

From a graph point of view, clustering is equivalent to breaking the graph into connected components (disjoint connected subgraphs), one for each cluster. Likewise, many clustering issues can be cast in graph-theoretic terms, e.g., the issues of cluster cohesion and the degree of coupling with other clusters can be measured by the number and strength of links between and within clusters. Also, many clustering techniques, e.g., single link and complete link (see Sec. 2.10), are most naturally described using graph representations.



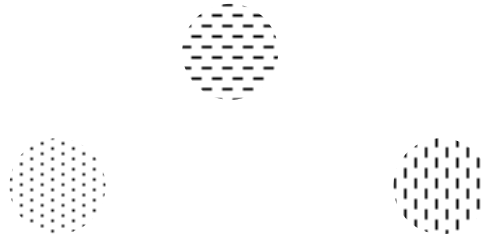
**Figure 2.** Four points, their proximity graph, and their corresponding data and proximity (distance) matrices.

## 2.6 Some Working Definitions of a Cluster

As mentioned above, the term, cluster, does not have a precise definition. However, several working definitions of a cluster are commonly used and are given below. There are two aspects of clustering that should be mentioned in conjunction with these definitions. First, clustering is sometimes viewed as finding only the most “tightly” connected points while discarding “background” or noise points. Second, it is sometimes acceptable to produce a set of clusters where a true cluster is broken into several subclusters (which are often combined later, by another technique). The key requirement in this latter situation is that the subclusters are relatively “pure,” i.e., most points in a subcluster are from the same “true” cluster.

- 1) **Well-Separated Cluster Definition:** A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any

point not in the cluster. Sometimes a threshold is used to specify that all the points in a cluster must be sufficiently close (or similar) to one another.



**Figure 3:** Three well-separated clusters of 2 dimensional points.

However, in many sets of data, a point on the edge of a cluster may be closer (or more similar) to some objects in another cluster than to objects in its own cluster. Consequently, many clustering algorithms use the following criterion.

- 2) **Center-based Cluster Definition:** A cluster is a set of objects such that an object in a cluster is closer (more similar) to the “center” of a cluster, than to the center of any other cluster. The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the “most representative” point of a cluster.



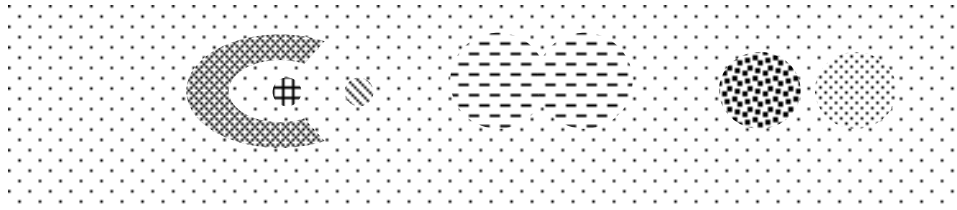
**Figure 4:** Four center-based clusters of 2 dimensional points.

- 3) **Contiguous Cluster Definition (Nearest Neighbor or Transitive Clustering):** A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.



**Figure 5:** Eight contiguous clusters of 2 dimensional points.

- 4) **Density-based definition:** A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density. This definition is more often used when the clusters are irregular or intertwined, and when noise and outliers are present. Notice that the contiguous definition would find only one cluster in Figure 6. Also note that the three curves don’t form clusters since they fade into the noise, as does the bridge between the two small circular clusters.



**Figure 6:** Six dense clusters of 2 dimensional points.

- 5) **Similarity-based Cluster definition:** A cluster is a set of objects that are “similar”, and objects in other clusters are not “similar.” A variation on this is to define a cluster as a set of points that together create a region with a uniform local property, e.g., density or shape.

## 2.7 Measures (Indices) of Similarity and Dissimilarity

The notion of similarity and dissimilarity (distance) seems fairly intuitive. However, the quality the quality of a cluster analysis depends critically on the similarity measure that is used and, as a consequence, hundreds of different similarity measures have been developed for various situations. The discussion here is necessarily brief.

### 2.7.1 Attribute types and Scales

The proximity measure (and the type of clustering used) depends on the attribute type and scale of the data. The three typical types of attributes are shown in Table 1, while the common data scales are shown in Table 2.

<b>Binary</b>	Two values, e.g., true and false.
<b>Discrete</b>	A finite number of values, or integers, e.g., counts.
<b>Continuous</b>	An effectively infinite number of real values, e.g., weight.

**Table 1:** Different attribute types.

<b>Qualitative</b>	<b>Nominal</b>	The values are just different names, e.g., colors or zip codes.
	<b>Ordinal</b>	The values reflect an ordering, nothing more, e.g., good, better, best.
<b>Quantitative</b>	<b>Interval</b>	The difference between values is meaningful, i.e., a unit of measurement exists. For example, temperature on the Celsius or Fahrenheit scales.
	<b>Ratio</b>	The scale has an absolute zero so that ratios are meaningful. Examples are physical quantities such as electrical current, pressure, or temperature on the Kelvin scale.

**Table 2:** Different attribute scales.

## 2.7.2 Euclidean Distance and Some Variations

The most commonly used proximity measure, at least for ratio scales (scales with an absolute 0) is the Minkowski metric, which is a generalization of the distance between points in Euclidean space.

$$p_{ij} = \left( \sum_{k=1}^d |x_{ik} - x_{jk}|^r \right)^{1/r}$$

where,  $r$  is a parameter,  $d$  is the dimensionality of the data object, and  $x_{ik}$  and  $x_{jk}$  are, respectively, the  $k^{\text{th}}$  components of the  $i^{\text{th}}$  and  $j^{\text{th}}$  objects,  $x_i$  and  $x_j$ .

For  $r = 1$ , this distance is commonly known as the  $L_1$  norm or city block distance. If  $r = 2$ , the most common situation, then we have the familiar  $L_2$  norm or Euclidean distance. Occasionally one might encounter the  $L_{\max}$  norm ( $L_{\infty}$  norm), which represents the case  $r \rightarrow \infty$ . Figure 7 gives the proximity matrices for the  $L_1$ ,  $L_2$  and  $L_{\infty}$  distances, respectively, using the data matrix from Figure 2.

The  $r$  parameter should not be confused with the dimension,  $d$ . For example, Euclidean, Manhattan and supremum distances are defined for all values of  $d$ , 1, 2, 3, ..., and specify different ways of combining the differences in each dimension (attribute) into an overall distance.

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L2	p1	p2	p3	p4
p1	0.000	2.828	3.162	5.099
p2	2.828	0.000	1.414	3.162
p3	3.162	1.414	0.000	2.000
p4	5.099	3.162	2.000	0.000

L1	p1	p2	p3	p4
p1	0.000	4.000	4.000	6.000
p2	4.000	0.000	2.000	4.000
p3	4.000	2.000	0.000	2.000
p4	6.000	4.000	2.000	0.000

L $\infty$	p1	p2	p3	p4
p1	0.000	2.000	3.000	5.000
p2	2.000	0.000	1.000	3.000
p3	3.000	1.000	0.000	2.000
p4	5.000	3.000	2.000	0.000

**Figure 7.** Data matrix and the corresponding  $L_1$ ,  $L_2$ , and  $L_{\infty}$  proximity matrices.

Finally, note that various Minkowski distances are metric distances. In other words, given a distance function,  $dist$ , and three points  $\mathbf{a}$ ,  $\mathbf{b}$ , and  $\mathbf{c}$ , these distances satisfy the following three mathematical properties: reflexivity (  $dist(\mathbf{a}, \mathbf{a}) = 0$  ), symmetry (  $dist(\mathbf{a}, \mathbf{b}) = dist(\mathbf{b}, \mathbf{a})$  ), and the triangle inequality (  $dist(\mathbf{a}, \mathbf{c}) \leq dist(\mathbf{a}, \mathbf{b}) + dist(\mathbf{b}, \mathbf{a})$  ). Not all distances or similarities are metric, e.g., the Jaccard measure of the following section. This introduces potential complications in the clustering process since in such cases,  $\mathbf{a}$  similar (close) to  $\mathbf{b}$  and  $\mathbf{b}$  similar to  $\mathbf{c}$ , does not necessarily imply  $\mathbf{a}$  similar to  $\mathbf{c}$ . The concept based clustering, which we discuss later, provides a way of dealing with such situations.

### 2.7.3 Similarity Measures Between Binary Vectors

These measures are referred to as similarity coefficients [JD88], and typically have values between 0 (not at all similar) and 1 (completely similar). The comparison of two binary vectors,  $\mathbf{a}$  and  $\mathbf{b}$ , leads to four quantities:

- $N_{01}$  = the number of positions where  $\mathbf{a}$  was 0 and  $\mathbf{b}$  was 1
- $N_{10}$  = the number of positions where  $\mathbf{a}$  was 1 and  $\mathbf{b}$  was 0
- $N_{00}$  = the number of positions where  $\mathbf{a}$  was 0 and  $\mathbf{b}$  was 0
- $N_{11}$  = the number of positions where  $\mathbf{a}$  was 1 and  $\mathbf{b}$  was 1

Two common similarity coefficients between binary vectors are the simple matching coefficient (SMC) and the Jaccard coefficient.

$$\text{SMC} = (N_{11} + N_{00}) / (N_{01} + N_{10} + N_{11} + N_{00})$$

$$\text{Jaccard} = N_{11} / (N_{01} + N_{10} + N_{11})$$

For the following two binary vectors,  $\mathbf{a}$  and  $\mathbf{b}$  we get  $\text{SMC} = 0.7$  and  $\text{Jaccard} = 0.7$ .

$$\begin{aligned} \mathbf{a} &= \mathbf{1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \mathbf{b} &= \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1} \end{aligned}$$

Conceptually, SMC equates similarity with the total number of matches, while J considers only matches on 1's to be important. There are situations in which both measures are more appropriate. For example, if the vectors represent students' answers to a True-False test, then both 0-0 and 1-1 matches are important and these two students are very similar, at least in terms of the grades they will get. If instead, the vectors indicate particular items purchased by two shoppers, then the Jaccard measure is more appropriate since it would be odd to say that the purchasing behavior of two customers is similar, even though they did not buy any of the same items.

## 2.8 Hierarchical and Partitional Clustering

The main distinction in clustering approaches is between hierarchical and partitional approaches. Hierarchical techniques produce a nested sequence of partitions, with a single, all-inclusive cluster at the top and singleton clusters of individual points at the bottom. Each intermediate level can be viewed as combining (splitting) two clusters from the next lower (next higher) level. (Hierarchical clustering techniques that start with one large cluster and split it are termed "divisive," while approaches that start with clusters containing a single point, and then merge them are called "agglomerative.") While most hierarchical algorithms involve joining two clusters or splitting a cluster into two sub-clusters, some hierarchical algorithms join more than two clusters in one step or split a cluster into more than two sub-clusters.

Partitional techniques create a one-level (unnested) partitioning of the data points. If  $K$  is the desired number of clusters, then partitional approaches typically find all  $K$  clusters at once. Contrast this with traditional hierarchical schemes, which bisect a cluster to get two clusters or merge two clusters to get one. Of course, a hierarchical approach can be used to generate a flat partition of  $K$  clusters, and likewise, the repeated application of a partitional scheme can provide a hierarchical clustering.



There are also other important distinctions between clustering algorithms: Does a clustering algorithm cluster on all attributes simultaneously (polythetic) or use only one attribute at a time (monothetic)? Does a clustering technique use one object at a time (incremental) or does the algorithm require access to all objects (non-incremental)? Does the clustering method allow a cluster to belong to multiple clusters (overlapping) or does it assign each object to a single cluster (non-overlapping)? Note that overlapping clusters are not the same as fuzzy clusters, but rather reflect the fact that in many real situations, objects belong to multiple classes.

## 2.9 Specific Partitional Clustering Techniques: K-means

The K-means algorithm discovers  $K$  (non-overlapping) clusters by finding  $K$  centroids (“central” points) and then assigning each point to the cluster associated with its nearest centroid. (A cluster centroid is typically the mean or median of the points in its cluster and “nearness” is defined by a distance or similarity function.) Ideally the centroids are chosen to minimize the total “error,” where the error for each point is given by a function that measures the discrepancy between a point and its cluster centroid, e.g., the squared distance. Note that a measure of cluster “goodness” is the error contributed by that cluster. For squared error and Euclidean distance, it can be shown [And73] that a gradient descent approach to minimizing the squared error yields the following basic K-means algorithm. (The previous discussion still holds if we use similarities instead of distances, but our optimization problem becomes a maximization problem.)

### Basic K-means Algorithm for finding $K$ clusters.

1. Select  $K$  points as the initial centroids.
2. Assign all points to the closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids don’t change (or change very little).

K-means has a number of variations, depending on the method for selecting the initial centroids, the choice for the measure of similarity, and the way that the centroid is computed. The common practice, at least for Euclidean data, is to use the mean as the centroid and to select the initial centroids randomly.

In the absence of numerical problems, this procedure converges to a solution, although the solution is typically a local minimum. Since only the vectors are stored, the space requirements are  $O(m*n)$ , where  $m$  is the number of points and  $n$  is the number of attributes. The time requirements are  $O(I*K*m*n)$ , where  $I$  is the number of iterations required for convergence.  $I$  is typically small and can be easily bounded as most changes occur in the first few iterations. Thus, the time required by K-means is efficient, as well as simple, as long as the number of clusters is significantly less than  $m$ .

Theoretically, the K-means clustering algorithm can be viewed either as a gradient descent approach which attempts to minimize the sum of the squared error of each point from cluster centroid [And73] or as procedure that results from trying to model the data as a mixture of Gaussian distributions with diagonal covariance matrices [Mit97].

## 2.10 Specific Hierarchical Clustering Techniques: MIN, MAX, Group Average

In hierarchical clustering the goal is to produce a hierarchical series of nested clusters, ranging from clusters of individual points at the bottom to an all-inclusive cluster at the top. A diagram called a dendrogram graphically represents this hierarchy and is an inverted tree that describes the order in which points are merged (bottom-up, agglomerative approach) or clusters are split (top-down, divisive approach). One of the attractions of hierarchical techniques is that they correspond to taxonomies that are very common in the biological sciences, e.g., kingdom, phylum, genus, species, ... (Some cluster analysis work occurs under the name of “mathematical taxonomy.”) Another attractive feature is that hierarchical techniques do not assume any particular number of clusters. Instead, any desired number of clusters can be obtained by “cutting” the dendrogram at the proper level. Finally, hierarchical techniques are thought to produce better quality clusters [JD88].

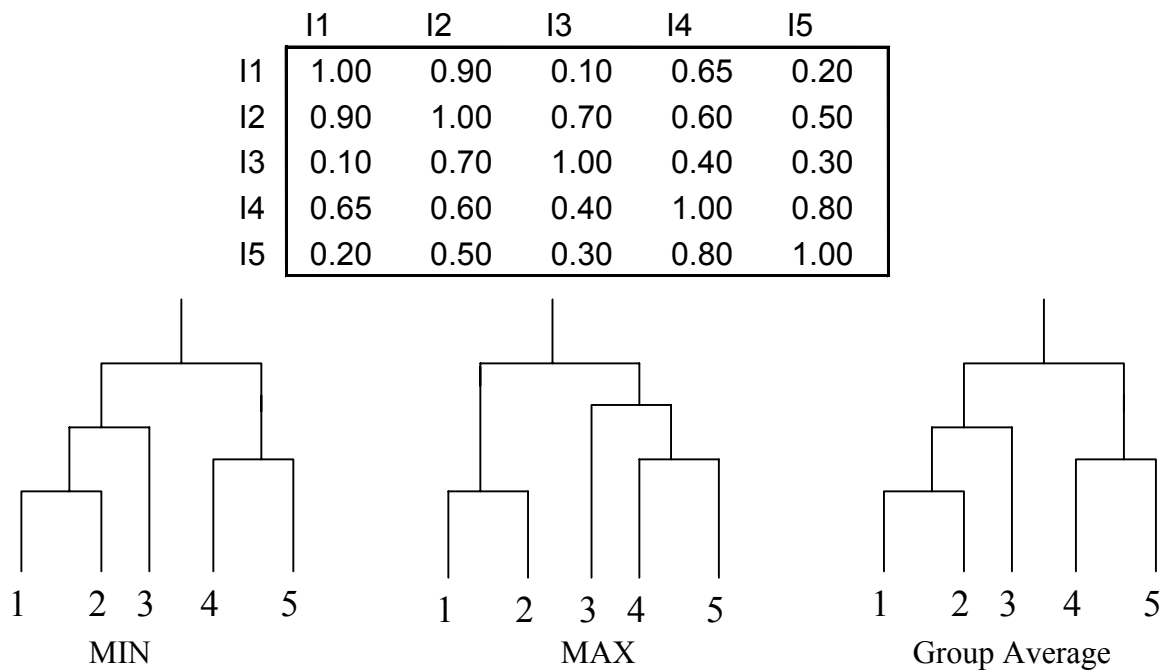
In this section we describe three agglomerative hierarchical techniques: MIN, MAX, and group average. For the single link or MIN version of hierarchical clustering, the proximity of two clusters is defined to be minimum of the distance (maximum of the similarity) between any two points in the different clusters. The technique is called single link, because if you start with all points as singleton clusters, and add links between points, strongest links first, these single links combine the points into clusters. Single link is good at handling non-elliptical shapes, but is sensitive to noise and outliers.

For the complete link or MAX version of hierarchical clustering, the proximity of two clusters is defined to be maximum of the distance (minimum of the similarity) between any two points in the different clusters. The technique is called complete link because, if you start with all points as singleton clusters, and add links between points, strongest links first, then a group of points is not a cluster until all the points in it are completely linked, i.e., form a clique. Complete link is less susceptible to noise and outliers, but can break large clusters, and has trouble with convex shapes.

For the group average version of hierarchical clustering, the proximity of two clusters is defined to be the average of the pairwise proximities between all pairs of points in the different clusters. Notice that this is an intermediate approach between MIN and MAX. This is expressed by the following equation:

$$proximity(cluster1, cluster2) = \frac{\sum_{\substack{p_1 \in cluster1 \\ p_2 \in cluster2}} proximity(p_1, p_2)}{size(cluster1) * size(cluster2)}$$

Figure 8 shows a table for a sample similarity matrix and three dendrograms, which respectively, show the series of merges that result from using the MIN, MAX, and group average approaches. In this simple case, MIN and group average produce the same clustering.



**Figure 8:** Dendograms produced by MIN, MAX and group average hierarchical clustering technique.

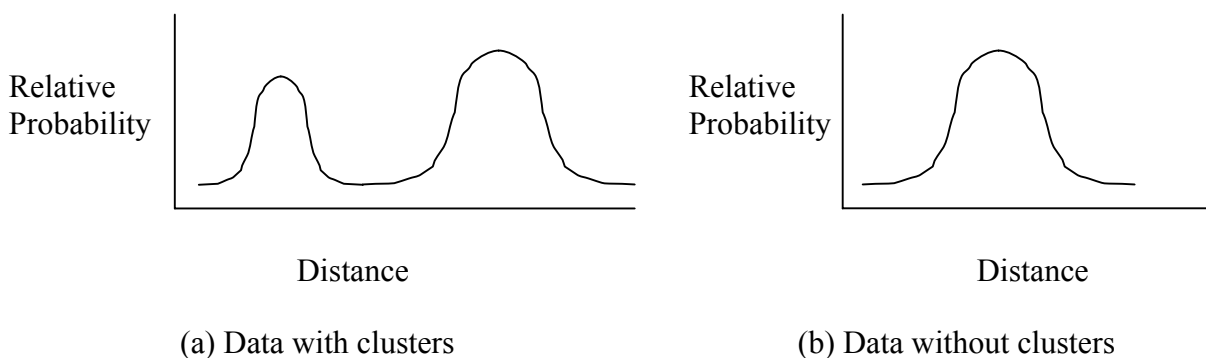
### 3 The “Curse of Dimensionality”

It was Richard Bellman who apparently originated the phrase, “the curse of dimensionality,” in a book on control theory [Bel61]. The specific quote from [Bel61], page 97, is “In view of all that we have said in the forgoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days.” The issue referred to in Bellman’s quote is the impossibility of optimizing a function of many variables by a brute force search on a discrete multidimensional grid. (The number of grids points increases exponentially with dimensionality, i.e., with the number of variables.) With the passage of time, the “curse of dimensionality” has come to refer to any problem in data analysis that results from a large number of variables (attributes).

In general terms, problems with high dimensionality result from the fact that a fixed number of data points become increasingly “sparse” as the dimensionality increase. To visualize this, consider 100 points distributed with a uniform random distribution in the interval  $[0, 1]$ . If this interval is broken into 10 cells, then it is highly likely that all cells will contain some points. However, consider what happens if we keep the number of points the same, but distribute the points over the unit square. (This corresponds to the situation where each point is two-dimensional.) If we keep the unit of discretization to be 0.1 for each dimension, then we have 100 two-dimensional cells, and it is quite likely that some cells will be empty. For 100 points and three dimensions, most of the 1000 cells

will be empty since there are far more points than cells. Conceptually our data is “lost in space” as we go to higher dimensions.

For clustering purposes, the most relevant aspect of the curse of dimensionality concerns the effect of increasing dimensionality on distance or similarity. In particular, most clustering techniques depend critically on the measure of distance or similarity, and require that the objects within clusters are, in general, closer to each other than to objects in other clusters. (Otherwise, clustering algorithms may produce clusters that are not meaningful.) One way of analyzing whether a data set may contain clusters is to plot the histogram (approximate probability density function) of the pairwise distances of all points in a data set (or of a sample of points if this requires too much computation.) If the data contains clusters, then the graph will typically show two peaks: a peak representing the distance between points in clusters, and a peak representing the average distance between points. Figures 9a and 9b, respectively, show idealized versions of the data with and without clusters. Also see [Bri95]. If only one peak is present or if the two peaks are close, then clustering via distance based approaches will likely be difficult. Note that clusters of different densities could cause the leftmost peak of Fig. 9a to actually become several peaks.



**Figure 9:** Plot of interpoint distances for data with and without clusters.

There has also been some work on analyzing the behavior of distances for high dimensional data. In [BGRS99], it is shown, for certain data distributions, that the relative difference of the distances of the closest and farthest data points of an independently selected point goes to 0 as the dimensionality increases, i.e.,

$$\lim_{d \rightarrow \infty} \frac{MaxDist - MinDist}{MinDist} = 0$$

For example, this phenomenon occurs if all attributes are i.i.d. (identically and independently distributed). Thus, it is often said, “in high dimensional spaces, distances between points become relatively uniform.” In such cases, the notion of the nearest neighbor of a point is meaningless. To understand this in a more geometrical way, consider a hyper-sphere whose center is the selected point and whose radius is the distance to the nearest data point. Then, if the relative difference between the distance to

nearest and farthest neighbors is small, expanding the radius of the sphere “slightly” will include many more points.

In [BGRS99] a theoretical analysis of several different types of distributions is presented, as well as some supporting results for real-world high dimensional data sets. This work was oriented towards the problem of finding the nearest neighbors of points, but the results also indicate potential problems for clustering high dimensional data.

The work just discussed was extended in [HAK00] to look at the absolute difference,  $\mathbf{MaxDist} - \mathbf{MinDist}$ , instead of the relative difference. It was shown that the behavior of the absolute difference between the distance to the closest and farthest neighbors of an independently selected point depends on the distance measure. In particular, for the  $L_1$  metric,  $\mathbf{MaxDist} - \mathbf{MinDist}$  increases with dimensionality, for the  $L_2$  metric,  $\mathbf{MaxDist} - \mathbf{MinDist}$  remains relatively constant, and for the  $L_d$  metric,  $d \geq 3$ ,  $\mathbf{MaxDist} - \mathbf{MinDist}$  goes to 0 as dimensionality increase. These theoretical results were also confirmed by experiments on simulated and real datasets. The conclusion is that the  $L_d$  metric,  $d \geq 3$ , is meaningless for high dimensional data.

The previous results indicate the potential problems with clustering high dimensional data sets, at least in cases where the data distribution causes the distances between points to become relatively uniform. However, things are sometimes not as bad as they might seem, for it is often possible to reduce the dimensionality of the data without losing important information. For example, sometimes it is known *a priori* that only a smaller number of variables are of interest. If so, then these variables can be selected, and the others discarded, thus reducing the dimensionality of the data set. More generally, data analysis (clustering or otherwise) is often preceded by a “feature selection” step that attempts to remove “irrelevant” features. This can be accomplished by discarding features that show little variation or which are highly correlated with other features. (Feature selection is a complicated subject in its own right.)

Another approach is to project points from a higher dimensional space to a lower dimensional space. The idea here is that that often data can be approximated reasonably well even if only a relatively small number of dimensions are kept, and thus, little “true” information is lost. Indeed, such techniques can, in some cases, enhance the data analysis because they are effective in removing noise. Typically this type of dimensionality reduction is accomplished by applying techniques from linear algebra or statistics such as Principal Component Analysis (PCA) [JD88] or Singular Value Decomposition (SVD) [Str86].

To make this more concrete we briefly illustrate with SVD. (Mathematically less inclined readers can skip this paragraph without loss.) A singular value decomposition of an  $m$  by  $n$  matrix,  $M$ , expresses  $M$  as the sum of simpler rank 1 matrices as follows:

$$M = \sum_{i=1}^n s_i u_i v_i^T$$
, where  $s_i$ , a scalar, is the  $i^{\text{th}}$  singular value of  $M$ ,  $u_i$  is the  $i^{\text{th}}$  left singular vector, and  $v_i$  is the  $i^{\text{th}}$  right singular vector. All singular values beyond the first  $r$ , where  $r = \text{rank}(M)$  are 0 and all left (right) singular vectors are orthogonal to each other and are of unit length. A matrix can be approximated by omitting some of the terms of the series that correspond to non-zero singular values. (Singular values are non-negative and ordered by decreasing magnitude.) Since the magnitudes of these singular values often decrease rapidly, an approximation based on a relatively small number of singular values, e.g., 50 or 100 out of 1000, is often sufficient for a productive data analysis.

Furthermore, it is not unusual to see data analyses that take only the first few singular values.

However, both feature selection and dimensionality reduction approaches based on PCA or SVD may be inappropriate if different clusters lie in different subspaces. Indeed, we emphasize that for many high dimensional data sets it is likely that clusters lie only in subsets of the full space. Thus, many algorithms for clustering high dimensional data automatically find clusters in subspaces of the full space. One example of such a clustering technique is “projected” clustering [AWYPP99], which also finds the set of dimensions appropriate for each cluster during the clustering process. More techniques that find clusters in subspaces of the full space will be discussed in Section 4.

In summary, high dimensional data is not like low dimensional data and needs different approaches. The next section presents recent work to provide clustering techniques for high dimensional data. While some of this work represents different developments of a single theme, e.g., grid based clustering, there is considerable diversity, perhaps because of high dimensional data, like low dimensional data is highly varied.

## **4 Recent Work in Clustering High Dimensional Data**

### **4.1 Clustering via Hypergraph Partitioning**

Hypergraph-based clustering [HKKM97] is an approach to clustering in high dimensional spaces, which is based on hypergraphs. (This is also work of one of the authors (Kumar), but not our recent work on clustering referenced earlier, which comes later in this section.) Hypergraphs are an extension of regular graphs, which relax the restriction that an edge can only join two vertices. Instead an edge can join many vertices. Hypergraph-based clustering consists of the following steps:

- 1) Define the condition for connecting several objects (each object is a vertex of the hypergraph) by a hyperedge.
- 2) Define a measure for the strength or weight of a hyperedge.
- 3) Use a graph-partitioning algorithm [KK98] to partition the hypergraph into two parts in such a way that the weight of the hyperedges cut is minimized.
- 4) Continue the partitioning until a fixed number of partitions are achieved, or until a new partition would produce a poor cluster, as measured by some fitness criteria.

In [HKKM97], the data being clustered is “market basket” data. With this kind of data there are a number of items and a number of “baskets”, or transactions, each of which contains a subset of all possible items. (A prominent example of market basket data is the subset of store items (products) purchased by customers in individual transactions – hence the name market basket data.) This data can be represented by a set of (very sparse) binary vectors – one for each transaction. Each item is associated with a dimension (variable), and a value of 1 indicates that the item was present in the transaction, while a value of 0 indicates that the item was not present.

The individual items are the vertices of the hypergraph. The hyperedges are determined by determining subsets of items that frequently occur together. For example, baby formula and diapers are often purchased together. These subsets of frequently co-

occurring items are called frequent itemsets and can be found using relatively simple and efficient algorithms [AS97].

The strength of the hyperedges is determined in the following manner. If the frequent itemset being considered is of size  $n$ , and the items of the frequent itemset are  $i_1, i_2 \dots i_n$ , then the strength of a hyperedge is obtained as follows:

- 1) Consider each individual item,  $i_j$ , in the frequent itemset.
- 2) Determine what fraction of the market baskets (transactions) that contain the other  $n - 1$  items also contain  $i_j$ . This (estimate of the) conditional probability that  $i_j$  occurs when the other items do is a measure of the strength of the association between the items.
- 3) Average these conditional probabilities together.

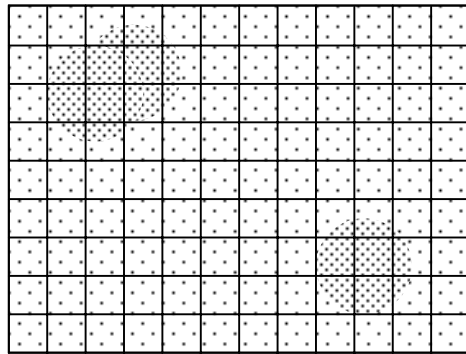
More formally the strength of a hyperedge is given by 
$$\frac{1}{n} \sum_{j=1}^n \text{prob}(i_j | i_1, \dots, i_{j-1}, i_{j+1}, i_n)$$

An important feature of this algorithm is that it transforms a problem in a sparse, high dimensional data space into a well-studied graph-partitioning problem that can be efficiently solved.

## 4.2 Grid Based Clustering Approaches

In its most basic form, grid based clustering is relatively simple:

- a) **Divide the space over which the data ranges into (hyper) rectangular cells**, e.g., by partitioning the range of values in each dimension into equally sized cells. See figure 10 for a two dimensional example of such a grid



**Figure 10:** Two dimensional grid for cluster detection

- b) **Discard low-density grid cells.** This assumes a density based definition of clusters, i.e., that high-density regions represent clusters, while low-density regions represent noise. This is often a good assumption, although density based approaches may have trouble when there are clusters are of widely differing densities.
- c) **Combine adjacent high-density cells to form clusters.** If high-density regions are adjacent, then they are joined to form a single cluster.

Assigning points to cells requires only linear time, i.e., the time complexity is  $O(n)$ , where  $n$  is the number of data points. (However, if the data is high dimensional or some dimensions have a large range, it is necessary to use data structures, e.g., hash tables [CLR90], that do not explicitly store the non-empty cells.) Discarding low-density cells also requires only linear time, at least if only non-empty cells are stored. However,

combining dense cells can potentially take  $O(n^2)$  time, since it may be necessary to compare each non-empty cell to every other. Nonetheless, if the number of dense grid cells is  $O(\sqrt{n})$ , then this step will also be linear.

There are a number of obvious concerns about grid-based clustering methods. The grids are square or rectangular and don't necessarily fit the shape of the clusters. This can be handled by increasing the number of grid cells, but at the price of increasing the amount of work, and if the grid size is halved the number of cells increases by a factor of  $2^d$ , where  $d$  is the number of dimensions. Also, since the density of a real cluster may vary, making the grid size too small might put "holes" in the cluster, especially with a small number of points. Finally, grid based clustering typically assumes that the distance between points is measure by and  $L_1$  or  $L_2$  distance measure.

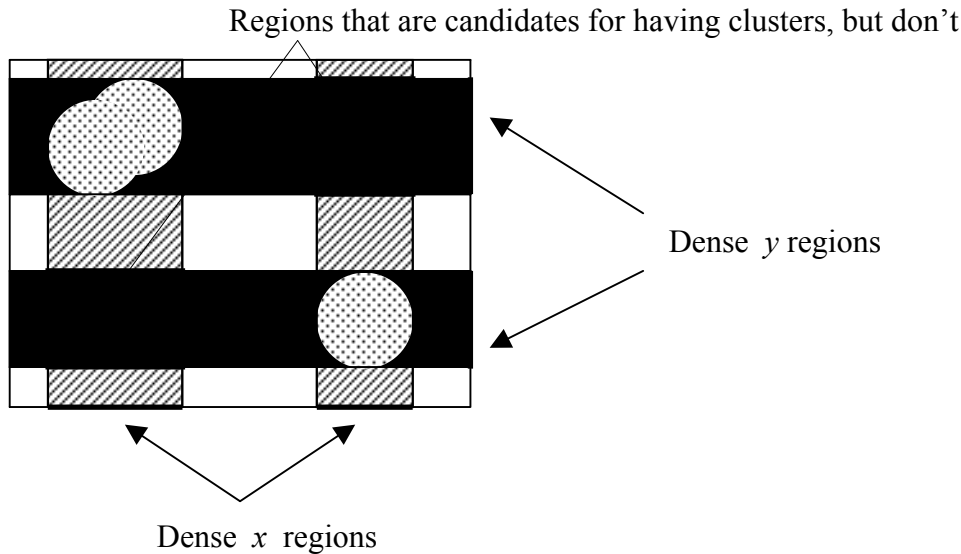
Also, despite the appealing efficiency of grid based clustering schemes, there are serious problems as the dimensionality of the data increases. First, the number of cells increases exponentially with increasing dimensionality. For example, even if each dimension is only split in two, there will still be  $2^d$  cells. Given 30 dimensional data, a grid based clustering approach will use, at least conceptually, a minimum of a billion cells. (Again by using algorithms for hash tables or sparse arrays, at most  $n$  cells need to be physically represented.) For all but the largest data sets, most of these cells will be empty. More importantly, it is very possible - particularly with a regular grid - that that a cluster might be divided into a large number of cells and that many or even all these cells might have a density less than the threshold.

Another problem is finding clusters in the full-dimensional space. To see this imagine that each point in one of the clusters in figure 10 is augmented with many additional variables, but that the values assigned to points in these dimensions are uniformly randomly distributed. Then almost every point will fall into a separate cell in the new, high dimensional space. Thus, as previously mentioned, clusters of points may only exist in subsets of high dimensional spaces. Of course, the number of possible subspaces is also exponential in the dimensionality of the space, yet another aspect of the curse of dimensionality.

### 4.2.1 CLIQUE

CLIQUE [AGGR98] is a clustering algorithm that attempts to deal with these problems and whose approach is based on the following interesting observation: a region that is dense in a particular subspace must create dense regions when projected onto lower dimensional subspaces. For example, if we examine the distribution of the  $x$  (horizontal) and  $y$  (vertical) coordinates of the points in Figure 11, we see dense regions in the one-dimensional distributions which reflect the existence two-dimensional clusters. In Figure 11, the gray horizontal columns and the slashed vertical columns indicate the projections of the clusters onto the vertical and horizontal axes, respectively. Figure 11 also illustrates that high density in a lower dimension can only suggest possible locations of clusters in a higher dimension, as the higher dimensional region formed by the intersection of two dense lower dimensional dense regions may not correspond to an actual cluster.



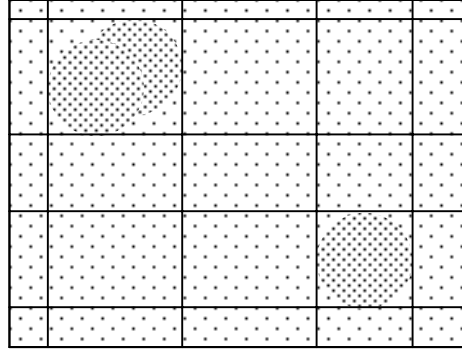


**Figure 11:** Illustration of the idea that density in high dimensions implies density in low dimensions, but not vice-versa.

However, by starting with dense one-dimensional intervals, it is possible to find the potential dense two-dimensional intervals, and by inspecting these, to find the actual dense two-dimensional intervals. This procedure can be extended to find dense units in any subspace, and to find them much more efficiently than by forming the cells corresponding to all possible subsets of dimensions and then searching for the dense units in these cells. However, CLIQUE still needs to use heuristics to reduce the subsets of dimensions investigated and the complexity of CLIQUE, while linear in the number of data points, is not linear in the number of dimensions.

#### 4.2.2 MAFIA

MAFIA (Merging Adaptive Finite Intervals And is more than a clique) [NGC99], which is a refinement of the CLIQUE approach, finds better clusters and achieves higher efficiency by using non-uniform grid cells. Specifically, rather than arbitrarily splitting the data into a pre-determined number of evenly spaced intervals, MAFIA partitions each dimension using a variable number of “adaptive intervals”, which better reflect the distribution of the data in that dimension. To illustrate, CLIQUE would more likely use a grid like that shown in Figure 10, and thus, would break each of the dense one-dimensional intervals into a number of subintervals, including a couple (at each end) that are of lesser density because they include part of the non-dense region. Conceptually, MAFIA starts with a large number of small intervals for each dimension and then combines adjacent intervals of similar density to end up with a smaller number of larger intervals. Thus, a MAFIA grid would likely look more like the idealized grid shown in Figure 12 than the suboptimal grid of figure 10.



**Figure 12:** MAFIA grid for our data.

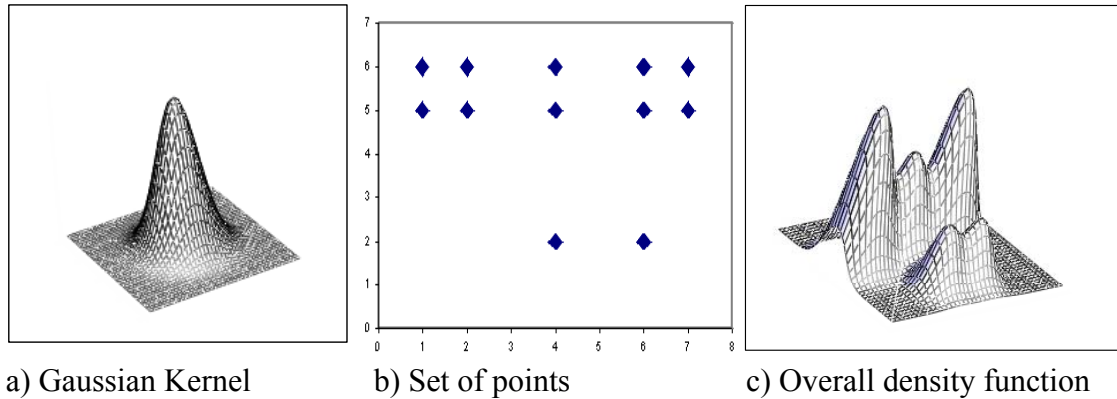
### 4.2.3 DENCLUE

A different approach to the same problem is provided by the DENCLUE [HK98]. We describe this approach in some detail, since this approach can be viewed as a generalization of other density-based approaches such as DBSCAN [EK SX96] and K-means. DENCLUE (DENSity CLUstEring) is a density clustering approach that takes a more formal approach to density based clustering by modeling the overall density of a set of points as the sum of “influence” functions associated with each point. The resulting overall density function will have local peaks, i.e., local density maxima, and these local peaks can be used to define clusters in a straightforward way. Specifically, for each data point, a hill climbing procedure finds the nearest peak associated with that point, and the set of all data points associated with a particular peak (called a local density attractor) becomes a (center-defined) cluster. However, if the density at a local peak is too low, then the points in the associated cluster are classified as noise and discarded. Also, if a local peak can be connected to a second local peak by a path of data points, and the density at each point on the path is above a minimum density threshold,  $\xi$ , then the clusters associated with these local peaks are merged. Thus, clusters of any shape can be discovered.

DENCLUE is based on a well-developed area of statistics and pattern recognition which is known as “kernel density estimation” [DHS00]. The goal of kernel density estimation (and many other statistical techniques as well) is to describe the distribution of the data by a function. For kernel density estimation, the contribution of each point to the overall density function is expressed by an “influence” (kernel) function. The overall density is then merely the sum of the influence functions associated with each point.

Typically the influence or kernel function is symmetric (the same in all directions) and its value (contribution) decreases as the distance from the point increases. For

example, for a particular point,  $x$ , the Gaussian function,  $K(x) = e^{\frac{-\text{distance}(x,y)^2}{2\sigma^2}}$ , is often used as a kernel function. ( $\sigma$  is a parameter which governs how quickly the influence of point drops off.) Figure 13a shows how a Gaussian function would look for a single two-dimensional point, while 13c shows what the overall density function produced by the Gaussian influence functions of the set of points shown in 13b.



**Figure 13:** Example of the Gaussian influence (kernel) function and an overall density function. ( $\sigma = 0.75$ ).

The DENCLUE algorithm has two steps, a preprocessing step and a clustering step. In the pre-clustering step, a grid for the data is created by dividing the minimal bounding hyper-rectangle into  $d$ -dimensional hyper-rectangles with edge length  $2\sigma$ . The grid cells that contain points are then determined. (As mentioned earlier, only the occupied grid cells need be constructed.) The grid cells are numbered with respect to a particular origin (at one edge of the bounding hyper-rectangle and these keys are stored in a search tree to provide efficient access in later processing. For each stored grid cell, the number of points, the sum of the points in the cell, and connections to neighboring cells are also stored.

For the clustering step DENCLUE, considers only the highly populated grid cells and the cells that are connected to them. For each point,  $x$ , the local density function is calculated only by considering those points that are from grid cells that are “close” to the point. As mentioned above, DENCLUE discards clusters associated with a density attractor whose density is less than  $\xi$ . Finally, DENCLUE merges density attractors that can be joined by a path of points, all of which have a density greater than  $\xi$ .

DENCLUE can be parameterized so that it behaves much like DBSCAN, but it is much more efficient than DBSCAN. DENCLUE can also behave like K-means by choosing  $\sigma$  appropriately and by omitting the step that merges center-defined clusters into arbitrary shaped clusters. Furthermore, by performing repeated clusterings for different values of  $\sigma$ , a hierarchical clustering can be obtained.

#### 4.2.4 OptiGrid

Despite the appealing characteristics of DENCLUE in low dimensional space, it does not work well as the dimensionality increase or if noise is present. Thus, the same researchers who created DENCLUE created OptiGrid. [HK99]. In this paper, the authors also make a number of interesting observations about the behavior of points in high dimensional space. First, they observe that for high dimensional data noise seems to correspond to uniformly distributed data in that it tends to produce data where there is only one point in a grid cell. More “centralized” distributions, like the Gaussian distribution result in far more cases where a grid cell has more than one point. Thus, the statistics of how many cells are multiply occupied can give us an idea of the amount of

noise in the data. Also, the authors provide additional comments on the observation that interpoint distances become relatively uniform as dimensionality increases. In particular, they point out that this means that the maximum density of a group of points may occur in a region of relatively empty space, a phenomenon known as the “empty point phenomenon.”

A fair amount of theoretical justification is presented in [HK99], but we will simplify our description. First, this will make the general approach easier to understand, since this simplification will be more in line with the description of the algorithms given above. Secondly, the algorithm actually implemented used the simplified approach.

- 1) For each dimension:
  - a) Generate a histogram of the data values. Note that this is equivalent to counting the points in a uniform one-dimensional grid (or set of intervals) imposed on the values.
  - b) Determine the noise level. This can be done by manually inspecting the histogram, if the dimensionality is not too high, but otherwise needs to be automated. For the results presented in the paper, the authors choose the manual approach.
  - c) Find the leftmost and rightmost maxima and the  $q-1$  maxima in between them. ( $q$  is the number of partitions of the data that we seek, and all these partitions could be in one dimension.)
  - d) Choose the  $q$  minima between the maxima found in the previous step. These points represent locations for possible cuts, i.e., locations where a hyperplane could be placed to partition the data. Choosing a low-density cell minimizes the chance of cutting through a cluster. However, it is not useful to cut at the edge of the data, and that is the reason for not choosing a minima at the edge, i.e., further right than the rightmost maxima or further left than the leftmost maxima.
  - e) Score each potential cut, e.g., by its density.
- 2) From all of the dimensions, select the best  $q$  cuts, i.e., the lowest density cuts.
- 3) Using these cuts, create a grid that partitions the data.
- 4) Find the highly populated grid cells and add them to the list of clusters.
- 5) Refine the list of clusters.
- 6) Repeat steps 1-5 using each cluster.

The key simplification that we made in the description and that was made in the implementation in the paper was that the separating hyperplanes must be parallel to some axis. To allow otherwise introduces additional time and coding complexity. The authors also show that using rectangular grids does not result in too much error, particularly as dimensionality increases.

In summary, OptiGrid seems a lot like MAFIA in that it creates a grid by using a data dependent partitioning. However, unlike MAFIA and CLIQUE, it does not face the problem of combinatorial search for the best subspace to use for partitioning. OptiGrid simply looks for the best cutting planes and creates a grid that is not likely to cut any clusters. It then locates potential clusters among this set of grid cells and further partitions them if possible. From an efficiency point, this is much better.

However, some details of the implementation of OptiGrid were vague, and there are a number of choices for parameters, e.g., how many cuts should be made. While OptiGrid seems promising, it should be remarked that another clustering approach, PDDP

[SB01], clusters data by making one optimal hyperplane cut at a time. (This approach is more computationally expensive than Optigrad.) One might think that such an approach would be able to match the best behavior of OptiGrid, but it has been shown that this method does not perform much better than a K-means approach. (Actually a combined approach is suggested in [SB01].) Thus, more evaluation is needed.

## 4.3 Noise Modeling in Wavelet Space

### 4.3.1 WaveCluster

WaveCluster [SCZ98] is a clustering technique that interprets the original data as a two-dimensional signal and then applies signal processing techniques (the wavelet transform) to map the original data to a new space where cluster identification is more straightforward. More specifically, WaveCluster defines a uniform two-dimensional grid on the data and represents the points in each grid cell by the number of points. Thus, a collection of two-dimensional data points becomes an image, i.e., a set of “gray-scale” pixels, and the problem of finding clusters becomes one of image segmentation.

While there are a number of techniques for image segmentation, wavelets have a couple of features that make them an attractive choice. First, the wavelet approach naturally allows for a multiscale analysis, i.e., the wavelet transform allows features, and hence, clusters, to be detected at different scales, e.g., fine, medium, and coarse. Secondly, the wavelet transform naturally lends itself to noise elimination.

The basic algorithm of WaveCluster is as follows:

- 1) **Create a grid and assign each data object to a cell in the grid.** The grid is uniform, but the grid size will vary for different scales of analysis. Each grid cell keeps track of the statistical properties of the points in that cell, but for wave clustering, only the number of points in the cell is used.
- 2) **Transform the data to a new space by applying the wavelet transform.** This results in 4 “subimages” at several different levels of resolution, an “average” image, an image that emphasizes the horizontal features, an image that emphasizes vertical features, and an image that emphasizes corners.
- 3) **Find the connected components in the transformed space.** The average subimage is used to find connected clusters, which are just groups of connected “pixels,” i.e., pixels which are connected to one another horizontally, vertically, or diagonally.
- 4) **Map the cluster labels of points in the transformed space back to points in the original space.** WaveCluster creates a lookup table that associates each point in the original with a point in the transformed space. Assignment of cluster labels to the original points is then straightforward.

In summary, the key features of WaveCluster are order independence, no need to specify a number of clusters (although it is helpful to know this in order to figure out the right scale to look at, speed (linear), the elimination of noise and outliers, and the ability to find arbitrarily shaped clusters. While the WaveCluster approach can theoretically be extended to more than two dimensions, it seems unlikely that WaveCluster will work well (efficiently and effectively) for medium or high dimensions.

### **4.3.2 Overcoming the Curse of Dimensionality via the Wavelet Transform**

The technique given in [MSB00] provides an approach for converting almost any kind of data to a gridded framework where a wavelet transform can be applied. The key idea is to treat the data matrix as an image matrix. A data matrix is a two dimensional array and so is an image matrix, and so, superficially, this is workable. However, the order of the rows and columns in a data matrix is arbitrary, i.e., they can be shuffled without changing the meaning of the data, while in an image the order is critical because of the spatial (sequential) relationship implied. Meaningful application of the wavelet transform depends on this spatial ordering, and thus, to treat a data array as an image requires the imposition of a meaningful order relationship on the rows (objects) and columns (variables) of the data matrix. This is accomplished by the use of matrix reordering techniques to permute the rows and columns to a standard form, which gathers larger or non-zero values towards the diagonal.

Once the matrix has been reordered, the data matrix is analyzed as if it were an image. In particular, the wavelet coefficients for each point are calculated for a variety of scales, e.g., 5 scales which differ by a factor of two. Thus, the original image is decomposed into 6 images (the image at 5 resolutions and a residual image.) Since most data has a lot of noise, statistical tests, which are based on an assumed statistical model for the noise in the data, can be applied to these wavelet coefficients to determine which ones are significant in a statistical sense. By setting all significant wavelet coefficients to 1, and each non-significant coefficient to 0, a binarized view of the data at each level of resolution can be obtained. By looking at either the binarized view or the original wavelet transformed view at the different levels, it is often possible to visually identify various clusters for further investigation.

Of course, the matrix reordering is an approximate process and may not always give exactly the same reordering from one run to the next. However, the authors indicate that this method is intended for quick exploratory clustering and show that it works reasonably well for some examples that they present.

### **4.4 A “Concept-Based” Approach to Clustering High Dimensional Data**

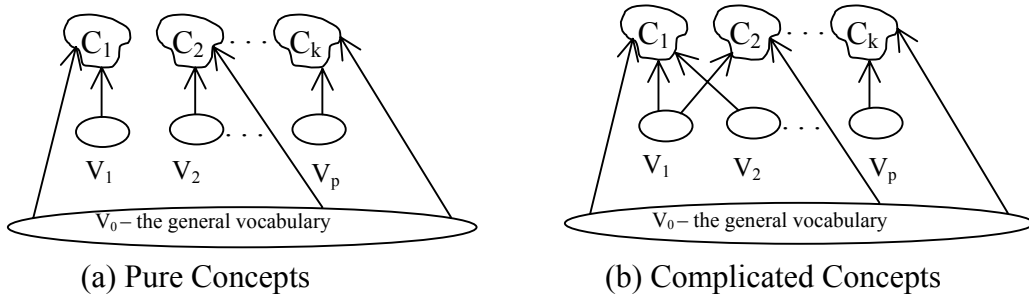
A key feature of some high dimensional data is that two objects may be highly similar even though commonly applied distance or similarity measures indicate that they are dissimilar or perhaps only moderately similar [GRS99]. Conversely, and perhaps more surprisingly, it is also possible that an object’s nearest or most similar neighbors may not be as highly “related” to the object as other objects which are less similar. To deal with this issue we have extended previous approaches that define the distance or similarity of objects in terms of the number of nearest neighbors that they share. The resulting approach defines similarity not in terms of shared attributes, but rather in terms of a more general notion of shared concepts. The rest of this section details our work in finding clusters in these “concept spaces,” and in doing so, provides a contrast to the approaches of the previous section, which were oriented to finding clusters in more traditional vector spaces.

### 4.4.1 Concept Spaces

For our purposes, a concept will be a set of attributes. As an example, for documents a concept would be a set of words that characterize a theme or topic such as “Art” or “Finance.” The importance of concepts is that, for many data sets, the objects in the data set can be viewed as being generated from one or more sets of concepts in a probabilistic way. Thus, a concept-oriented approach to documents would view each document as consisting of words that come from one or more concepts, i.e., sets of words or vocabularies, with the probability of each word being determined by an underlying statistical model. We refer to data sets with this sort of structure as concept spaces, even though the underlying data may be represented as points in a vector space or in some other format. The practical relevance of concept spaces is that data belonging to concept spaces must be treated differently in terms of how the similarity between points should be calculated and how the objects should be clustered.

To make this more concrete we detail a concept-based model for documents. Figure 14a shows the simplest model, which we call the “pure concepts” model. In this model, the words from a document in the  $i^{th}$  class,  $C_i$ , of documents come from either the general vocabulary,  $V_0$ , or from exactly one of the specialized vocabularies,  $V_1, V_2, \dots, V_p$ . For this model the vocabularies are just sets of words and possess no additional structure. In this case, as in the remaining cases discussed, all vocabularies can overlap. Intuitively, however, a specialized word that is found in a document is more likely to have originated from a specialized vocabulary than from the general vocabulary.

Figure 14b is much like Figure 14a and shows a slightly more complicated (realistic) model, which we call the “multiple concepts” model. The only difference from the previous model is that a word in a document from a particular class may come from more than one specialized vocabulary. More complicated models are also possible.



**Figure 14:** Different concept models.

A statistical model for the concept-based models shown above could be the following. A word,  $w$ , in a document,  $d$ , from a cluster  $C_i$  comes with one or more vocabularies with a probability given by  $P(w | C_i) = \sum P(w | V_j) * P(V_j | C_i)$ . For the pure concepts model, each word of a document comes only from the general vocabulary and one of the specialized vocabularies. For the multiple concepts model, each word of a document comes from one or more specialized vocabularies.

### 4.4.2 Problems with Similarity in Concept Spaces

In the beginning of this section, it was mentioned that similarity measures might behave in unexpected ways in concept spaces. We present some examples and discussion to indicate why this is so.

In the following we are assuming that the variables are what are sometimes called “unary” variables, i.e., it makes sense to say that an object has that attribute or doesn’t have that attribute. For example, a document may or may not contain a certain word, or a customer may or may not purchase a certain item. Counts, categorical attributes, or binary attributes can be easily translated into unary attributes, but the situation is more complicated with most continuous attributes. We omit discussion of such cases to keep the explanations simple.

Our first example is similar to one in [GRS99]. Consider a concept space where all the objects fall into two groups, A and B. Objects from group A are generated by selecting three of the attributes (with equal probability) from the concept set {1, 2, 3, 4, 5} and objects from group B are generated by selecting three of the attributes from the concept set {4, 5, 6, 7, 8}. Suppose that we have generated the following three objects  $x = \{1, 2, 3\}$ ,  $y = \{3, 4, 5\}$ , and  $z = \{4, 5, 6\}$ . (We can also represent these points as binary vectors, e.g.,  $x = (1\ 1\ 1\ 0\ 0\ 0\ 0\ 0)$ .) Clearly, points  $x$  and  $y$  belong to group A, while point  $z$  belongs to group B. However, just as clearly, most similarity measures, e.g., the Jaccard measure, would judge points  $y$  and  $z$  to be most similar, as they share two out of their three attributes, while  $x$  and  $y$  share only one attribute.

### 4.4.3 The need for indirect similarity in concept spaces

If we carefully examine document sets, we observe that the average similarity between documents within a cluster (using the popular cosine measure) is almost always lower than 0.6, and it generally lies between 0.2 and 0.5. This means that, on the average, two documents in the same cluster share about 20% - 50% of their terms (assuming binary attributes). If a documents’ similarity with its nearest neighbor is 0.3, then we should not put the two documents in the same cluster right away. We should notice that the similarity between the two is actually low. Consider the set of documents in Table 3.

A	1	1	1	0	0	0	0	0	0	0	0
B	0	1	1	1	0	0	0	0	0	0	0
C	1	1	0	1	1	1	1	1	0	0	0
D	0	0	0	1	1	1	1	1	0	1	1
E	0	0	0	0	0	0	0	1	1	1	0
F	0	0	0	0	0	0	0	0	1	1	1

**Table 3:** Sample set of document

The most similar two documents are C & D, but the appropriate clusters for this set are A, B, C and D, E, F. In both of the clusters, every document shares 2 attributes with any other document. First 4 attributes bind A, B and C together, while the last 4 bind D, E and F together.



A document cluster should contain documents that form a topic, and this does not imply placing the closest neighbor of a document in the same cluster as we have seen in the previous example. If we look at the indirect similarities; number of length 2 links between documents, we will see that C & D have only one indirect link while A-B, A-C and B-C will all have 2 indirect links. Hence, A-B-C and D-E-F form coherent clusters.

For a more realistic example, consider actual similarity measures for documents. Documents are represented using the vector-space model [Rij79], where each document,  $d$ , is considered to be a vector,  $\mathbf{d}$ , in the term-space (set of document “words”). In its simplest form, each document is represented by the (TF) vector,

$$\mathbf{d}_{tf} = (tf_1, tf_2, \dots, tf_n),$$

where  $tf_i$  is the frequency of the  $i^{th}$  term in the document. (Normally very common words are stripped out completely and different forms of a word are reduced to one canonical form.) In addition, we use the version of this model that weights each term based on its *inverse document frequency* (IDF) in the document collection. (This discounts frequent words with little discriminating power.) Finally, in order to account for documents of different lengths, each document vector is normalized so that it is of unit length.

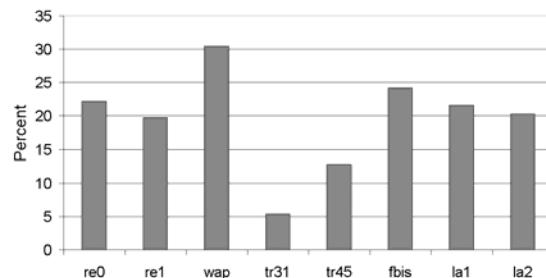
There are a number of possible measures for computing the similarity between documents, but the most common one is the cosine measure, which is defined as

$$\text{cosine}(\mathbf{d}_1, \mathbf{d}_2) = (\mathbf{d}_1 \bullet \mathbf{d}_2) / \|\mathbf{d}_1\| \|\mathbf{d}_2\|,$$

where  $\bullet$  indicates the vector dot product and  $\|\mathbf{d}\|$  is the length of vector  $\mathbf{d}$ . Notice that this measure is similar to the Jaccard measure in that it only considers the presence of terms to be important.

As mentioned above, what distinguishes documents of different classes is the frequency with which words are used. In particular, each class typically has a “core” vocabulary of words that are used more frequently. For example, documents about finance will often talk about money, mortgages, trade, etc., while documents about sports talk about players, coaches, games, etc. These core vocabularies may overlap, documents may use more than one “core” vocabulary, and any particular document may contain words from these different “core” vocabularies, even if it does not belong to the class of documents that typically uses such words.

Each document has only a subset of all words from the complete vocabulary. Thus, because of the probabilistic nature of how words are distributed, any two documents may share many of the same words. Thus, it should not be surprising that two documents can often be nearest neighbors without belonging to the same class. For a variety of document datasets (see [SKK00]). Figure 15 shows the percentage of documents whose nearest neighbor is not of the same class. (Classes were pre-assigned, for example, by using the section of the newspaper in which a document occurred.)



**Figure 15:** Percent nearest neighbors of a different class.

the section of the newspaper in which a document occurred.)

Since hierarchical and K-means clustering, which are often used for document clustering, use the cosine measure to decide how to cluster documents, they will inevitably make mistakes. In particular, agglomerative hierarchical clustering will often put documents of the same class in the same cluster at the earliest stages of the clustering process. Because of the way that hierarchical clustering works, these “mistakes” cannot be fixed once they happen. K-means can potentially do better, because it continually reassigns documents to the most appropriate cluster as the clustering proceeds. However, K-means is still based on a definition of similarity that is suspect, and we have observed that clusters produced by K-means often contain documents that don’t have a consistent topic.

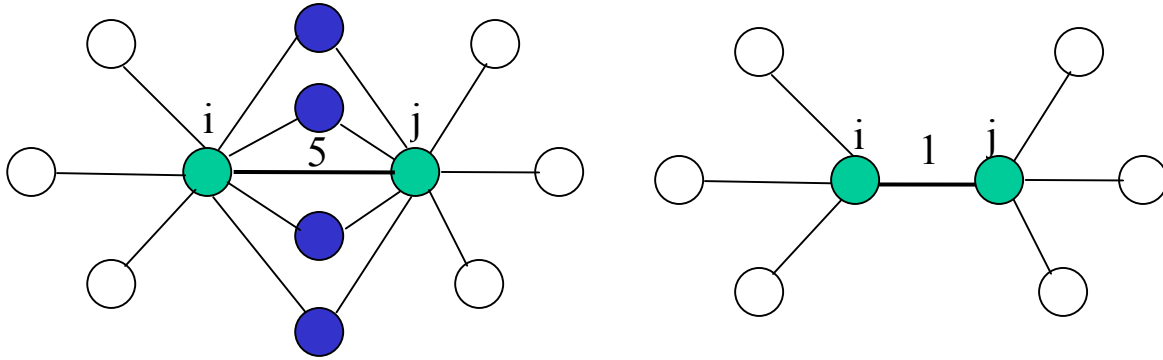
In cases where nearest neighbors are unreliable, a different approach is needed that relies on more global properties. We discuss a general approach based on nearest neighbors, and then discuss our own approach.

#### **4.4.4 A Shared Nearest Neighbor Approach to Similarity**

Our clustering algorithm is based on a shared nearest neighbor clustering algorithm described in [JP73]. A similar approach, but for hierarchical clustering, was developed in [GK78]. Recently, a couple of other clustering algorithms have used shared nearest neighbor ideas [GRS99, KHK99].

We explain the approach of [JP73], which we call Jarvis-Patrick clustering, in more detail since it is the basis for our clustering technique. We will describe the shared nearest neighbor algorithm in [JP73] using graph terminology. (Recall that from a graph point of view, clustering is equivalent to breaking the graph into connected components, one for each cluster.)

- 1) First the  $n$  nearest neighbors of all points are found. In graph terms this can be regarded as breaking all but the  $n$  strongest links from a point to other points in the proximity graph. This forms what we call a “nearest neighbor graph.” Note that the nearest neighbor graph is just a sparsified version of the original similarity graph, where we break the links to less similar points.
- 2) We then determine the number of nearest neighbors shared by any two points. In graph terminology we form what we call the “shared nearest neighbor” graph. We do this by replacing each link (in the nearest neighbor graph) between two points by the number of neighbors that the points share. In other words [GRS99], this is the number of length 2 paths between any two points in the nearest neighbor graph. In the Fig. 16 the links between nodes (documents) indicate that they are similar (direct similarity). The numbers show the strength of the link in the shared nearest neighbor graph.



**Figure 16:** Illustration of the ways points can share neighbors.

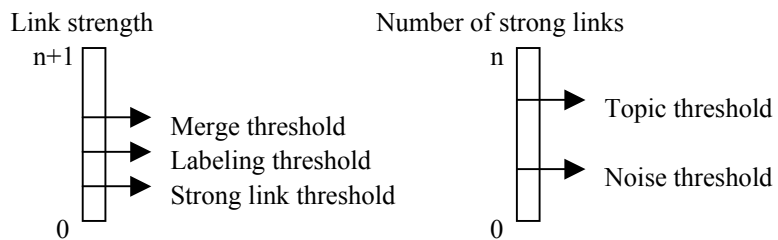
- 3) All pairs of points are compared and if any two points share more than  $T$  neighbors, i.e., have a link in the shared nearest neighbor graph with a weight more than our threshold value,  $T$  ( $T \leq n$ ), then the two points and any cluster they are part of are merged. In other words, clusters are connected components in the shared nearest neighbor graph after we sparsify using a threshold.

This approach has a number of nice properties. It can handle clusters of different densities since the shared nearest neighbor approach is self-scaling. Also, this approach is transitive, i.e., if point,  $p$ , shares lots of nearest neighbors with point,  $q$ , which in turn shares lots of nearest neighbors with point,  $r$ , then points  $p$ ,  $q$  and  $r$  all belong to the same cluster. The transitive property, in turn, allows this technique to handle clusters of different sizes and shapes. As described in the next sections, we have extended the Jarvis-Patrick approach.

#### 4.4.5 Our Clustering Approach

We begin by calculating the document similarity matrix, i.e., the matrix which gives the cosine similarity for each pair of documents. Once this similarity matrix is calculated, we find the first  $n$  nearest neighbors for each document. (Every object is considered to be its own 0<sup>th</sup> neighbor.) In the nearest neighbor graph, there is a link from object  $i$  to object  $j$ , if  $i$  and  $j$  both have each other in their nearest neighbor list. In the shared nearest neighbor graph, there is a link from  $i$  to  $j$  if there is a link from  $i$  to  $j$  in the nearest neighbor graph and the strength of this link is equal to the number of shared nearest neighbors of  $i$  and  $j$ .

At this point, we could just apply a threshold, and take all the connected components of the shared nearest neighbor graph as our final clusters [JP73]. However, this threshold would need to be set too high since this is a single link approach and



**Figure 16:** Different types of parameters.

will give poor results when patterns in the dataset are not very significant. On the other hand, when a high threshold is applied, a natural cluster will be split into many small clusters due to the variations in the similarity within the cluster. We address these problems with the clustering algorithm described below.

There are two types of parameters used in this algorithm: one type relates to the strength of the links in the shared nearest neighbor graph, the other type relates to the number of strong links for an object. If the strength of a link is greater than a threshold, then that link is labeled as a strong link.

The details of our shared nearest neighbor clustering algorithm are as follows:

- 1) For every point  $i$  in the dataset, calculate the connectivity,  $\text{conn}[i]$ , the number of strong links the point has.
- 2) For a point  $i$  in the dataset, if  $\text{conn}[i] < \text{noise threshold}$ , then that point is not considered in the clustering since it is similar to only a few of its neighbors. Similarly, if  $\text{conn}[i] > \text{topic threshold}$ , then that point is similar to most of its neighbors and is chosen to represent its neighborhood.
- 3) For any pair of points  $(i, j)$  in the dataset, if  $i$  and  $j$  share significant numbers of their neighbors, i.e. the strength of the link between  $i$  and  $j$  is greater than the *merge threshold*, then they will appear together in the final clustering if either one of them (or both) is chosen to be a representative. Our algorithm will not suffer from the effects of transitivity since every other point on a chain of links has to be chosen to be a representative. In other words, two objects that are not directly related will be put in the same cluster only if there are many other objects between them that are connected with strong links, half of which must represent their own neighborhood.
- 4) Labeling step: Having defined the representative points and the points strongly related to them, we can bring back some of the points that did not survive the *merge threshold*. This is done by scanning the shared nearest neighbor list of all the points that are part of a cluster, and checking whether those points have links to points that don't belong to any cluster and have a link strength greater than the labeling threshold.

After applying the algorithm described above, there may be singleton clusters. These singleton clusters are *not* equivalent to the singleton clusters obtained using the JP method. Note that if only a threshold is applied after converting the nearest neighbor graph to the shared nearest neighbor graph, there will be several clusters (which are the connected components after applying the threshold), and the rest will be singletons. By introducing the topic threshold, we are able to mark the documents that have similar documents around. In the end, if a document that is labeled as a topic remains as a singleton, this does not mean that it is a noise document. For that document to be labeled as a topic, it must have enough number of strong links, which means that it has many similar neighbors but the strength of those links were not strong enough to merge them.

Singleton clusters give us some idea about the less dominant topics in the dataset, and they are far more valuable than the singletons that are left out (labeled as background). To the best of our knowledge, there is no other algorithm that produces valuable singleton (or very small) clusters. Being able to make use of the singleton clusters can be very useful. If we're trying to detect topics in a document set, we don't have to force the parameters of the algorithms to the edge to find out small topics. If we

end up getting a singleton cluster, that document will give us an idea about several other documents, whereas noise documents do not give us any idea about any other document.

The method described above finds communities of objects, where an object in a community shares a certain fraction of its neighbors with at least some number of neighbors. While the probability of an object belonging to a class different from its nearest neighbor's class may be relatively high, this probability decreases as the two objects share more and more neighbors. This is the main idea behind the algorithm.

#### **4.4.6 An Application of Concept-based Clustering to Documents**

We illustrate concept based clustering by considering clustering for documents. Given a set of documents, clustering is often used to group the documents, in the hope that each such group will represent documents with a common theme or topic (concept). Initially hierarchical clustering was used to cluster documents [EW89]. This approach has the advantage of producing a set of nested document clusters, which can be interpreted as a topic hierarchy or tree, from general to more specific topics. In practice, while the clusters at different levels of the hierarchy sometimes represent documents with consistent concepts or topics, it is common for many clusters to be a mixture of topics, even at lower, more refined levels of the hierarchy. More recently, as document collections have grown larger, K-means clustering has emerged as a more efficient approach to producing clusters of documents [DM00, KH00, SKK00]. K-means clustering produces a set of un-nested clusters, and the top (most frequent or highest "weight") terms of the cluster are used to characterize the topic of the cluster. Once again it is not unusual for some clusters to be mixtures of topics.

By applying our algorithm for clustering concept-based data to documents, we have created an approach that more consistently produces clusters of documents with strong, coherent themes (concepts), even though many documents may be omitted in the process. After all, in an arbitrary collection of documents, e.g., a set of newspaper articles, there is no reason to expect that all documents belong to a group with a strong topic or theme. While a concept-based approach does not provide a complete organization of all documents, it does identify the "nuggets" of information in a document collection and might profitably be applied to practical problems such as grouping the search results of a Web search engine.

#### **4.4.7 Sample Results for Concept-based Clustering of Documents**

We applied our technique to the data set LA1, which is from the Los Angeles Times data of TREC-5. (See [ESK01] for more details.). The words in Table 4 are the most important (frequent) 6 words in each document cluster. In Table 4 we see that all the documents in the first cluster are related to NCAA, while all the documents in the second cluster are related to NBA. Even though both sets of documents are basketball related, our clustering algorithm found them as separate clusters. We ran the K-means algorithm on the same dataset, and interestingly, all of the documents in these two clusters appeared in the same K-means cluster together with a number of documents related to gymnastics, swimming, as well as several apparently unrelated documents. The reason that K-means put all these sports documents in the same cluster is that sports documents tend to share a lot of common words such as, 'score,' 'half,' 'quarter,' 'game,' 'ball,' etc. This example indicates that pair-wise similarity by itself isn't a good measure for clustering documents.

The NCAA cluster					
wolfpack	towson	lead	tech	Scor	North
syracus	scor	georgia	dome	auburn	Louisvill
Scor	lead	throw	half	Free	Iowa
Scor	Fresno	unlv	lead	lockhart	jacksonvil
Panther	pittsburgh	sooner	brookin	Scor	Game
Iowa	minnesota	scor	illinoi	wisconsin	Burton
Scor	half	virginia	georgetown	lead	Kansa
Burson	louisvill	scor	ohio	game	Ellison

The NBA cluster					
Pacer	scor	piston	shot	game	hawkin
Cavali	mckei	charlott	scor	superson	cleveland
Scor	game	tripucka	basket	hornet	straight
levingston	hawk	jordan	malon	buck	quarter
daugherti	piston	warrior	cavali	shot	Eject

**Table 4:** Six most important words in document cluster.

Thus, using an approach based on shared nearest neighbors (SNN), we can get purer clusters, although not all the documents are assigned to clusters. However, in order to make a fair comparison, we decided to remove from K-means clusters all documents that were far away from the centroid of their cluster. We observed that this improved the misclassification rate only slightly. Finally, we also noticed [ESK01], when we looked at the individual documents in a ‘supposedly poor’ SNN cluster, that the documents did form a coherent group even though they have different class labels.

#### 4.4.8 Some Final Comments on Concept Based Clustering

While we have restricted our discussion here to concept based clustering for documents, the shared nearest neighbor approach to similarity on which it is based can be applied to many different sorts of data. In particular, the shared nearest neighbor approach from which concept-based is derived, was originally used for two-dimensional spatial data, and we have also successfully applied our data to such data. A major task ahead of us is to more precisely define those situations in which is it applicable.

## 5 Conclusions

In this paper we have provided a brief introduction to cluster analysis with an emphasis on the challenge of clustering high dimensional data. The principal challenge in extending cluster analysis to high dimensional data is to overcome the “curse of dimensionality,” and we described, in some detail, the way in which high dimensional data is different from low dimensional data, and how these differences might affect the process of cluster analysis. We then described several recent approaches to clustering high dimensional data, including our own work on concept-based clustering. All of these approaches have been successfully applied in a number of areas, although there is a need for more extensive study to compare these different techniques and better understand their strengths and limitations.

In particular, there is no reason to expect that one type of clustering approach will be suitable for all types of data, even all high dimensional data. Statisticians and other data analysts are very cognizant of the need to apply different tools for different types of data, and clustering is no different.

Finally, high dimensional data is only one issue that needs to be considered when performing cluster analysis. In closing we mention some other, only partially resolved, issues in cluster analysis: scalability to large data sets, independence of the order of input, effective means of evaluating the validity of clusters that are produced, easy interpretability of results, an ability to estimate any parameters required by the clustering

technique, an ability to function in an incremental manner, and robustness in the presence of different underlying data and cluster characteristics.

## References

- [AGGR98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. “Automatic subspace clustering of high-dimensional data for data mining applications,” In ACM SIGMOD Conference on Management of Data (1998).
- [AS97] Rakesh Agrawal and Ramakrishnan Srikant, “Fast Algorithms for Mining Association Rules,” In Proceedings of the 20<sup>th</sup> VLDB Conference, pp. 487-499, Santiago, Chile (1997).
- [AWYPP99] Charu Aggarwal, Cecilia Procopiuc, Joel Wolf, Phillip Yu, and Jong Park. “Fast algorithms for projected clustering,” In ACM SIGMOD Conference, (1999).
- [And73] Michael R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York and London (1973).
- [Bel61] R. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, New Jersey (1961).
- [Bri95] Sergey Brin, “Near Neighbor Search in Large Metric Spaces,” Proceedings of the 21st International Conference on Very Large Databases (VLDB-1995), pp. 574-584, Zurich Switzerland, Morgan Kaufmann (1995).
- [BGRS99] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft, (1998), “When is ‘nearest neighbor’ meaningful?” In Proceedings of 7th International Conference on Database Theory (ICDT-1999), Jerusalem, Israel, pp. 217-235, (1999).
- [CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, Prentice Hall, 1990.
- [DHS00] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Recognition*, Wiley and Sons (2000).
- [DM00] Inderjit S. Dhillon and Dharmendra S. Modha, "Concept Decompositions for Large Sparse Text Data using Clustering," *Machine Learning*, 42(1), pp. 143-175, January (2001).
- [Don00] David L. Donoho, “High Dimensional Data Analysis: The Curses and Blessings of Dimensionality,” American Math. Society Conference: Mathematical Challenges of the 21st Century, Los Angeles, CA, August, 6-11 (2000). (Currently only available on the Web at <http://www-stat.stanford.edu/~donoho/Lectures/AMS2000/AMS2000.html>)
- [EK SX96] Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD 96), Portland, Oregon, pp. 226-231 (1996).
- [ESK01] Levent Ertöz, Michael Steinbach, and Vipin Kumar “Finding Topics in Collections of Documents: A Shared Nearest Neighbor Approach,” In Proceeding of Text Mining Workshop, First International SIAM Data Mining Conference, Chicago, IL (2001).

- [EW89] A. El-Hamdouchi and P. Willet, "Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval," *The Computer Journal*, Vol. 32, No. 3, (1989).
- [FR98] Chris Fraley and Adrian E. Raferty, *How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis*, Technical Report No. 329, Department of Statistics, University of Washington, Seattle, Washington (1998).
- [GK78] K. C. Gowda and G. Krishna, "Agglomerative Clustering Using the Concept of Mutual Nearest Neighborhood," *Pattern Recognition*, Vol. 10, pp. 105-112 (1978).
- [GRS00] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," In Proceedings of the 15th International Conference on Data Engineering (ICDE '99), pp. 512-521 (1999).
- [HAK00] A. Hinneburg, C. Aggarwal, and D.A. Keim, "What is the nearest neighbor in high dimensional spaces?" In Proceedings 26th International Conference on Very Large Data Bases (VLDB-2000), Cairo, Egypt, September 2000, pp. 506-515, Morgan Kaufmann (2000).
- [HK98] Alexander Hinneburg and Daniel A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, pp. 58-65 (1998).
- [HK99] Alexander Hinneburg and Daniel A. Keim, "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering," In Proceedings of 25th International Conference on Very Large Data Bases (VLDB-1999), pp. 506-517, Edinburgh, Scotland, September, 1999, Morgan Kaufmann (1999).
- [HKKM97] Eui-Hong Han, George Karypis, Vipin Kumar, and Bomshad Mobasher, "Clustering In a High-Dimensional Space Using Hypergraph Models," Technical Report TR-97-063, Department of Computer Science, University of Minnesota, Minneapolis, Minnesota (1997).
- [HKKR99] Frank Hoppner, Frank Klawonn, Rudolf Kruse, and Thomas Runkler, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition*, John Wiley & Sons (1999).
- [JD88] Anil K. Jain and Richard C. Dubes, *Algorithms for Clustering Data*, Prentice Hall (1988).
- [JMF99] Anil K. Jain, M. N. Murty, P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, 31(3): 264-323 (1999).
- [JP73] R. A. Jarvis and E. A. Patrick, "Clustering Using a Similarity Measure Based on Shared Nearest Neighbors," *IEEE Transactions on Computers*, Vol. C-22, No. 11, November (1973).
- [KH00] George Karypis and Eui-Hong (Sam) Han, "Concept Indexing: A Fast Dimensionality Reduction Algorithm with Applications to Document Retrieval & Categorization", Ninth International Conference on Information and Knowledge Management (CIKM 2000), McLean, VA (2000).



- [KHK99] George Karypis, Eui-Hong Han, and Vipin Kumar, "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling," *IEEE Computer*, Vol. 32, No. 8, August, pp. 68-75 (1999).
- [KK98] George Karypis and Vipin Kumar *hMETIS 1.5: A hypergraph partitioning package*, Technical report, Department of Computer Science, University of Minnesota (1998).
- [KR90] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley and Sons (1990).
- [Mit97] Tom Mitchell, *Machine Learning*, McGraw Hill, New York (1997).
- [MSB00] Fionn. Murtagh, Jean-Luc Starck, and Michael W. Berry, "Overcoming the curse of dimensionality in clustering by means of the wavelet transform", *The Computer Journal*, 43, 107-120 (2000).
- [NGC99] Harasha Nagesh, Sanjay Goil, and Alok Choudhary, "MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets," Technical Report Number CPDC-TR-9906-019, Center for Parallel and Distributed Computing, Northwestern University (1999).
- [Rij79] C. J. Van Rijsbergen, *Information Retrieval*, Butterworth, London, second edition, (1979).
- [SB01] Sergio M. Savaresi and Daniel L. Boley, "On the Performance of Bisecting K-Means and PDDP," In Proceeding of the First International SIAM Data Mining Conference, Chicago, IL (2001).
- [Str86] Gilbert Strang. *Linear Algebra and its Applications*, Harcourt Brace Jovanovich, third edition (1986).
- [SCZ98] Gholamhosein Sheikholeslami, Surojit Chatterjee and Aidong Zhang "Wavecluster: A multi-resolution clustering approach for very large spatial databases," In Proceedings of the 24th VLDB Conference (1998).
- [SKK00] Michael Steinbach, George Karypis, and Vipin Kumar, "A Comparison of Document Clustering Algorithms," In Proceedings of the Text Mining Workshop for The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (KDD 2000), Boston, MA. (2000).