CHAPTER 5

Clustering

5.1 INTRODUCTION

Clustering is similar to classification in that data are grouped. However, unlike classification, the groups are not predefined. Instead, the grouping is accomplished by finding similarities between data according to characteristics found in the actual data. The groups are called *clusters*. Some authors view clustering as a special type of classification. In this text, however, we follow a more conventional view in that the two are different. Many definitions for clusters have been proposed:

- Set of like elements. Elements from different clusters are not alike.
- The distance between points in a cluster is less than the distance between a point in the cluster and any point outside it.

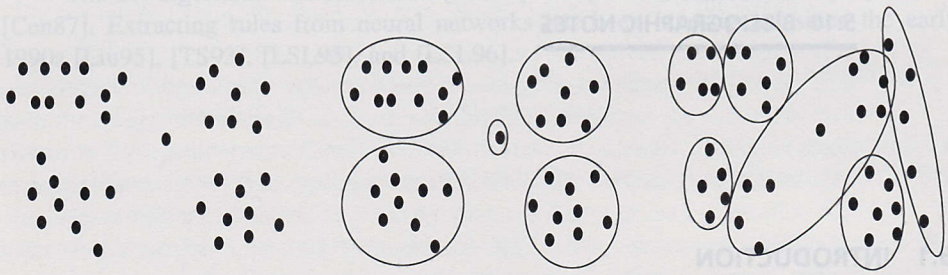
A term similar to clustering is *database segmentation*, where like tuples (records) in a database are grouped together. This is done to partition or segment the database into components that then give the user a more general view of the data. In this text, we do not differentiate between segmentation and clustering. A simple example of clustering is found in Example 5.1. This example illustrates the fact that determining how to do the clustering is not straightforward.

EXAMPLE 5.1

An international online catalog company wishes to group its customers based on common features. Company management does not have any predefined labels for these groups. Based on the outcome of the grouping, they will target marketing and advertising campaigns to the different groups. The information they have about the customers includes

TABLE 5.1: Sample Data for Example 5.1

Income	Age	Children	Marital Status	Education
$25,000	35	3	Single	High school
$15,000	25	1	Married	High school
$20,000	40	0	Single	High school
$30,000	20	0	Divorced	High school
$20,000	25	3	Divorced	College
$70,000	60	0	Married	College
$90,000	30	0	Married	Graduate school
$200,000	45	5	Married	Graduate school
$100,000	50	2	Divorced	College



(a) Group of homes (b) Geographic distance-based (c) Size-based

FIGURE 5.1: Different clustering attributes.

income, age, number of children, marital status, and education. Table 5 shows some tuples from this database for customers in the United States. Depending on the type of advertising, not all attributes are important. For example, suppose the advertising is for a special sale on children's clothes. We could target the advertising only to the persons with children. One possible clustering is that shown by the divisions of the table. The first group of people have young children and a high school degree, while the second group is similar but have no children. The third group has both children and a college degree. The last two groups have higher incomes and at least a college degree. The very last group has children. Different clusterings would have been found by examining age or marital status.

As illustrated in Figure 5.1, a given set of data may be clustered on different attributes. Here a group of homes in a geographic area is shown. The first type of clustering is based on the location of the home. Homes that are geographically close to each other are clustered together. In the second clustering, homes are grouped based on the size of the house.

Clustering has been used in many application domains, including biology, medicine, anthropology, marketing, and economics. Clustering applications include plant and animal

classification, disease classification, image processing, pattern recognition, and document retrieval. One of the first domains in which clustering was used was biological taxonomy. Recent uses include examining Web log data to detect usage patterns.

When clustering is applied to a real-world database, many interesting problems occur:

- Outlier handling is difficult. Here the elements do not naturally fall into any cluster. They can be viewed as solitary clusters. However, if a clustering algorithm attempts to find larger clusters, these outliers will be forced to be placed in some cluster. This process may result in the creation of poor clusters by combining two existing clusters and leaving the outlier in its own cluster.
- Dynamic data in the database implies that cluster membership may change over time.
- Interpreting the semantic meaning of each cluster may be difficult. With classification, the labeling of the classes is known ahead of time. However, with clustering, this may not be the case. Thus, when the clustering process finishes creating a set of clusters, the exact meaning of each cluster may not be obvious. Here is where a domain expert is needed to assign a label or interpretation for each cluster.
- There is no one correct answer to a clustering problem. In fact, many answers may be found. The exact number of clusters required is not easy to determine. Again, a domain expert may be required. For example, suppose we have a set of data about plants that have been collected during a field trip. Without any prior knowledge of plant classification, if we attempt to divide this set of data into similar groupings, it would not be clear how many groups should be created.
- Another related issue is what data should be used for clustering. Unlike learning during a classification process, where there is some a priori knowledge concerning what the attributes of each classification should be, in clustering we have no supervised learning to aid the process. Indeed, clustering can be viewed as similar to unsupervised learning.

We can then summarize some basic features of clustering (as opposed to classification):

- The (best) number of clusters is not known.
- There may not be any a priori knowledge concerning the clusters.
- Cluster results are dynamic.

The clustering problem is stated as shown in Definition 5.1. Here we assume that the number of clusters to be created is an input value, k. The actual content (and interpretation) of each cluster, K_j, $1 \leq j \leq k$, is determined as a result of the function definition. Without loss of generality, we will view that the result of solving a clustering problem is that a set of clusters is created: $K = \{K_1, K_2, \ldots, K_k\}$.

Definition 5.1. Given a database $D = \{t_1, t_2, \ldots, t_n\}$ of tuples and an integer value k, the **clustering problem** is to define a mapping $f : D \rightarrow \{1, \ldots, k\}$ where each t_i is assigned to one cluster K_j, $1 \leq j \leq k$. A **cluster**, K_j, contains precisely those tuples mapped to it; that is, $K_j = \{t_i \mid f(t_i) = K_j, 1 \leq i \leq n, \text{ and } t_i \in D\}$.

A classification of the different types of clustering algorithms is shown in Figure 5.2. Clustering algorithms themselves may be viewed as hierarchical or partitional. With

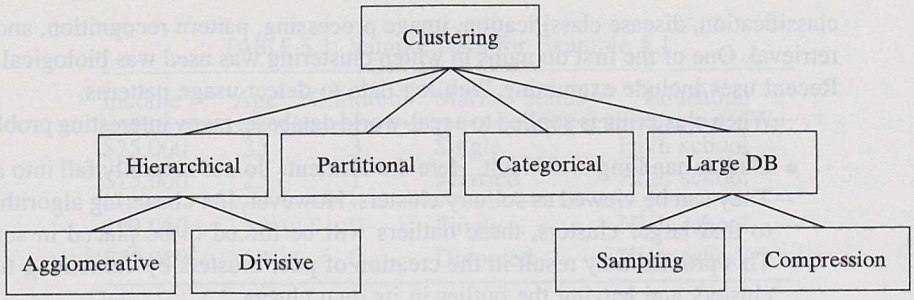


FIGURE 5.2: Classification of clustering algorithms.

hierarchical clustering, a nested set of clusters is created. Each level in the hierarchy has a separate set of clusters. At the lowest level, each item is in its own unique cluster. At the highest level, all items belong to the same cluster. With hierarchical clustering, the desired number of clusters is not input. With *partitional* clustering, the algorithm creates only one set of clusters. These approaches use the desired number of clusters to drive how the final set is created. Traditional clustering algorithms tend to be targeted to small numeric databases that fit into memory. There are, however, more recent clustering algorithms that look at categorical data and are targeted to larger, perhaps dynamic, databases. Algorithms targeted to larger databases may adapt to memory constraints by either sampling the database or using data structures, which can be compressed or pruned to fit into memory regardless of the size of the database. Clustering algorithms may also differ based on whether they produce overlapping or nonoverlapping clusters. Even though we consider only nonoverlapping clusters, it is possible to place an item in multiple clusters. In turn, nonoverlapping clusters can be viewed as extrinsic or intrinsic. *Extrinsic* techniques use labeling of the items to assist in the classification process. These algorithms are the traditional classification supervised learning algorithms in which a special input training set is used. *Intrinsic* algorithms do not use any a priori category labels, but depend only on the adjacency matrix containing the distance between objects. All algorithms we examine in this chapter fall into the intrinsic class.

The types of clustering algorithms can be furthered classified based on the implementation technique used. Hierarchical algorithms can be categorized as agglomerative or divisive. "*Agglomerative*" implies that the clusters are created in a bottom-up fashion, while *divisive* algorithms work in a top-down fashion. Although both hierarchical and partitional algorithms could be described using the agglomerative vs. divisive label, it typically is more associated with hierarchical algorithms. Another descriptive tag indicates whether each individual element is handled one by one, *serial* (sometimes called *incremental*), or whether all items are examined together, *simultaneous*. If a specific tuple is viewed as having attribute values for all attributes in the schema, then clustering algorithms could differ as to how the attribute values are examined. As is usually done with decision tree classification techniques, some algorithms examine attribute values one at a time, *monothetic*. *Polythetic* algorithms consider all attribute values at one time. Finally, clustering algorithms can be labeled based on the mathematical formulation given to the algorithm: graph theoretic or matrix algebra. In this chapter we generally use the graph approach and describe the input to the clustering algorithm as an adjacency matrix labeled with distance measures.

We discuss many clustering algorithms in the following sections. This is only a representative subset of the many algorithms that have been proposed in the literature. Before looking at these algorithms, we first examine possible similarity measures and examine the impact of outliers.

5.2 SIMILARITY AND DISTANCE MEASURES

There are many desirable properties for the clusters created by a solution to a specific clustering problem. The most important one is that a tuple within one cluster is more like tuples within that cluster than it is similar to tuples outside it. As with classification, then, we assume the definition of a similarity measure, $\text{sim}(t_i, t_l)$, defined between any two tuples, $t_i, t_l \in D$. This provides a more strict and alternative clustering definition, as found in Definition 5.2. Unless otherwise stated, we use the first definition rather than the second. Keep in mind that the similarity relationship stated within the second definition is a desirable, although not always obtainable, property.

DEFINITION 5.2. Given a database $D = \{t_1, t_2, \ldots, t_n\}$ of tuples, a similarity measure, $\text{sim}(t_i, t_l)$, defined between any two tuples, $t_i, t_l \in D$, and an integer value k, the **clustering problem** is to define a mapping $f : D \rightarrow \{1, \ldots, k\}$ where each t_i is assigned to one cluster K_j, $1 \leq j \leq k$. Given a cluster, K_j, $\forall t_{jl}, t_{jm} \in K_j$ and $t_i \notin K_j$, $\text{sim}(t_{jl}, t_{jm}) > \text{sim}(t_{jl}, t_i)$.

A distance measure, $\text{dis}(t_i, t_j)$, as opposed to similarity, is often used in clustering. The clustering problem then has the desirable property that given a cluster, K_j, $\forall t_{jl}$, $t_{jm} \in K_j$ and $t_i \notin K_j$, $\text{dis}(t_{jl}, t_{jm}) \leq \text{dis}(t_{jl}, t_i)$.

Some clustering algorithms look only at numeric data, usually assuming metric data points. *Metric* attributes satisfy the triangular inequality. The clusters can then be described by using several characteristic values. Given a cluster, K_m of N points $\{t_{m1}, t_{m2}, \ldots, t_{mN}\}$, we make the following definitions [ZRL96]:

$$\text{centroid} = C_m = \frac{\sum_{i=1}^{N}(t_{mi})}{N} \tag{5.1}$$

$$\text{radius} = R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{mi} - C_m)^2}{N}} \tag{5.2}$$

$$\text{diameter} = D_m = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(t_{mi} - t_{mj})^2}{(N)(N-1)}} \tag{5.3}$$

Here the centroid is the "middle" of the cluster; it need not be an actual point in the cluster. Some clustering algorithms alternatively assume that the cluster is represented by one centrally located object in the cluster called a *medoid*. The radius is the square root of the average mean squared distance from any point in the cluster to the centroid, and the diameter is the square root of the average mean squared distance between all pairs of points in the cluster. We use the notation M_m to indicate the medoid for cluster K_m.

Many clustering algorithms require that the distance between clusters (rather than elements) be determined. This is not an easy task given that there are many interpretations for distance between clusters. Given clusters K_i and K_j, there are several standard alternatives to calculate the distance between clusters. A representative list is:

- **Single link:** Smallest distance between an element in one cluster and an element in the other. We thus have $\text{dis}(K_i, K_j) = \min(\text{dis}(t_{il}, t_{jm}))\forall t_{il} \in K_i \notin K_j$ and $\forall t_{jm} \in K_j \notin K_i$.
- **Complete link:** Largest distance between an element in one cluster and an element in the other. We thus have $\text{dis}(K_i, K_j) = \max(\text{dis}(t_{il}, t_{jm}))\forall t_{il} \in K_i \notin K_j$ and $\forall t_{jm} \in K_j \notin K_i$.
- **Average:** Average distance between an element in one cluster and an element in the other. We thus have $\text{dis}(K_i, K_j) = \text{mean}(\text{dis}(t_{il}, t_{jm}))\forall t_{il} \in K_i \notin K_j$ and $\forall t_{jm} \in K_j \notin K_i$.
- **Centroid:** If clusters have a representative centroid, then the centroid distance is defined as the distance between the centroids. We thus have $\text{dis}(K_i, K_j) = \text{dis}(C_i, C_j)$, where C_i is the centroid for K_i and similarly for C_j.
- **Medoid:** Using a medoid to represent each cluster, the distance between the clusters can be defined by the distance between the medoids: $\text{dis}(K_i, K_j) = \text{dis}(M_i, M_j)$.

5.3 OUTLIERS

As mentioned earlier, *outliers* are sample points with values much different from those of the remaining set of data. Outliers may represent errors in the data (perhaps a malfunctioning sensor recorded an incorrect data value) or could be correct data values that are simply much different from the remaining data. A person who is 2.5 meters tall is much taller than most people. In analyzing the height of individuals, this value probably would be viewed as an outlier.

Some clustering techniques do not perform well with the presence of outliers. This problem is illustrated in Figure 5.3. Here if three clusters are found (solid line), the outlier will occur in a cluster by itself. However, if two clusters are found (dashed line), the two (obviously) different sets of data will be placed in one cluster because they are closer together than the outlier. This problem is complicated by the fact that many clustering algorithms actually have as input the number of desired clusters to be found.

Clustering algorithms may actually find and remove outliers to ensure that they perform better. However, care must be taken in actually removing outliers. For example, suppose that the data mining problem is to predict flooding. Extremely high water level values occur very infrequently, and when compared with the normal water level values may seem to be outliers. However, removing these values may not allow the data mining algorithms to work effectively because there would be no data that showed that floods ever actually occurred.

Outlier detection, or *outlier mining*, is the process of identifying outliers in a set of data. Clustering, or other data mining, algorithms may then choose to remove or treat these values differently. Some outlier detection techniques are based on statistical techniques. These usually assume that the set of data follows a known distribution and that outliers can be detected by well-known tests such as *discordancy tests*. However, these

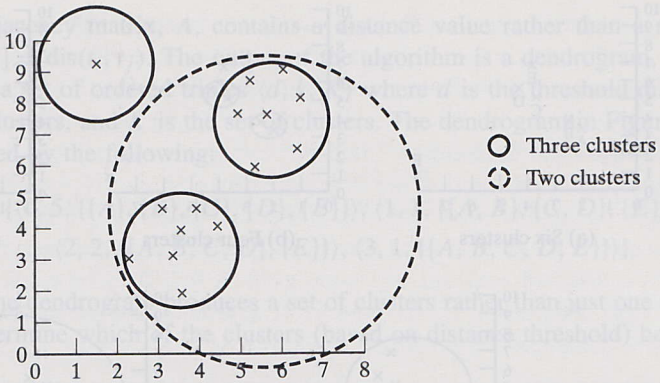


FIGURE 5.3: Outlier clustering problem.

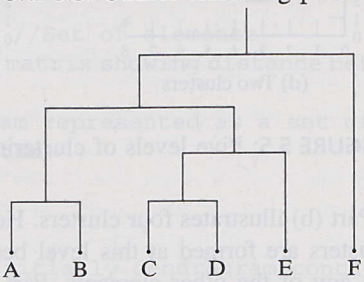


FIGURE 5.4: Dendrogram for Example 5.2.

tests are not very realistic for real-world data because real-world data values may not follow well-defined data distributions. Also, most of these tests assume a single attribute value, and many attributes are involved in real-world datasets. Alternative detection techniques may be based on distance measures.

5.4 HIERARCHICAL ALGORITHMS

As mentioned earlier, hierarchical clustering algorithms actually creates sets of clusters. Example 5.2 illustrates the concept. Hierarchical algorithms differ in how the sets are created. A tree data structure, called a *dendrogram*, can be used to illustrate the hierarchical clustering technique and the sets of different clusters. The root in a dendrogram tree contains one cluster where all elements are together. The leaves in the dendrogram each consist of a single element cluster. Internal nodes in the dendrogram represent new clusters formed by merging the clusters that appear as its children in the tree. Each level in the tree is associated with the distance measure that was used to merge the clusters. All clusters created at a particular level were combined because the children clusters had a distance between them less than the distance value associated with this level in the tree. A dendrogram for Example 5.2 is seen in Figure 5.4.

EXAMPLE 5.2

Figure 5.5 shows six elements, $\{A, B, C, D, E, F\}$, to be clustered. Parts (a) to (e) of the figure show five different sets of clusters. In part (a) each cluster is viewed to consist of