9

CLUSTER ANALYSIS

Chapter Objectives

- Distinguish between different representations of clusters and different measures of similarities.
- Compare the basic characteristics of agglomerative- and partitional-clustering algorithms.
- Implement agglomerative algorithms using single-link or complete-link measures of similarity.
- Derive the K-means method for partitional clustering and analysis of its complexity.
- Explain the implementation of incremental-clustering algorithms and its advantages and disadvantages.
- Introduce concepts of density clustering, and algorithms Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Balanced and Iterative Reducing and Clustering Using Hierarchies (BIRCH).
- Discuss why validation of clustering results is a difficult problem.

Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition. Mehmed Kantardzic.

Cluster analysis is a set of methodologies for automatic classification of samples into a number of groups using a measure of association so that the samples in one group are similar and samples belonging to different groups are not similar. The input for a system of cluster analysis is a set of samples and a measure of similarity (or dissimilarity) between two samples. The output from cluster analysis is a number of groups (clusters) that form a partition, or a structure of partitions, of the data set. One additional result of cluster analysis is a generalized description of every cluster, and this is especially important for a deeper analysis of the data set's characteristics.

9.1 CLUSTERING CONCEPTS

Organizing data into sensible groupings is one of the most fundamental approaches of understanding and learning. Cluster analysis is the formal study of methods and algorithms for natural grouping, or clustering, of objects according to measured or perceived intrinsic characteristics or similarities. Samples for clustering are represented as a vector of measurements, or more formally, as a point in a multidimensional space. Samples within a valid cluster are more similar to each other than they are to a sample belonging to a different cluster. Clustering methodology is particularly appropriate for the exploration of interrelationships among samples to make a preliminary assessment of the sample structure. Human performances are competitive with automatic-clustering procedures in one, two, or three dimensions, but most real problems involve clustering in higher dimensions. It is very difficult for humans to intuitively interpret data embedded in a high-dimensional space.

Table 9.1 shows a simple example of clustering information for nine customers, distributed across three clusters. Two features describe customers: The first feature is the number of items the customers bought, and the second feature shows the price they paid for each.

Customers in Cluster 1 purchase a few high-priced items; customers in Cluster 2 purchase many high-priced items; and customers in Cluster 3 purchase few low-priced

TABLE 9.1. Sample Set of Clusters Consisting of Similar Objects

	Number of Items	Price
Cluster 1	2	1700
	3	2000
	4	2300
Cluster 2	10	1800
	12	2100
	11	2500
Cluster 3	2	100
	3	200
	3	350

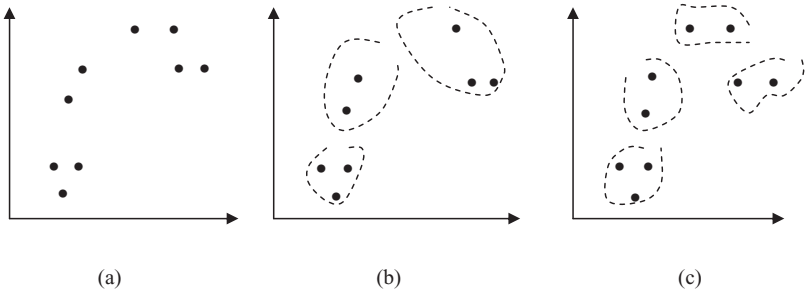


Figure 9.1. Cluster analysis of points in a 2D-space. (a) Initial data; (b) three clusters of data; (c) four clusters of data.

items. Even this simple example and interpretation of a cluster's characteristics shows that clustering analysis (in some references also called unsupervised classification) refers to situations in which the objective is to construct decision boundaries (classification surfaces) based on unlabeled training data set. The samples in these data sets have only input dimensions, and the learning process is classified as unsupervised.

Clustering is a very difficult problem because data can reveal clusters with different shapes and sizes in an n-dimensional data space. To compound the problem further, the number of clusters in the data often depends on the resolution (fine vs. coarse) with which we view the data. The next example illustrates these problems through the process of clustering points in the Euclidean two-dimensional (2-D) space. Figure 9.1a shows a set of points (samples in a 2-D space) scattered on a 2-D plane. Let us analyze the problem of dividing the points into a number of groups. The number of groups N is not given beforehand. Figure 9.1b shows the natural clusters bordered by broken curves. Since the number of clusters is not given, we have another partition of four clusters in Figure 9.1c that is as natural as the groups in Figure 9.1b. This kind of arbitrariness for the number of clusters is a major problem in clustering.

Note that the above clusters can be recognized by sight. For a set of points in a higher dimensional Euclidean space, we cannot recognize clusters visually. Accordingly, we need an objective criterion for clustering. To describe this criterion, we have to introduce a more formalized approach in describing the basic concepts and the clustering process.

An input to a cluster analysis can be described as an ordered pair (X, *s*), or (X, *d*), where X is a set of object descriptions represented with samples, and *s* and *d* are measures for similarity or dissimilarity (distance) between samples, respectively. Output from the clustering system is a partition $\Lambda = \{G_1, G_2, \ldots, G_N\}$, where G_k, $k = 1, \ldots, N$ is a crisp subset of X such that

$$G_1 \cup G_2 \cup \ldots \cup G_N = X, \quad \text{and}$$

$$G_i \cap G_j = \varnothing \quad \text{for } i \neq j$$

The members $G_1, G_2, \ldots, G_N$ of Λ are called clusters. Every cluster may be described with some characteristics. In discovery-based clustering, both the cluster (a

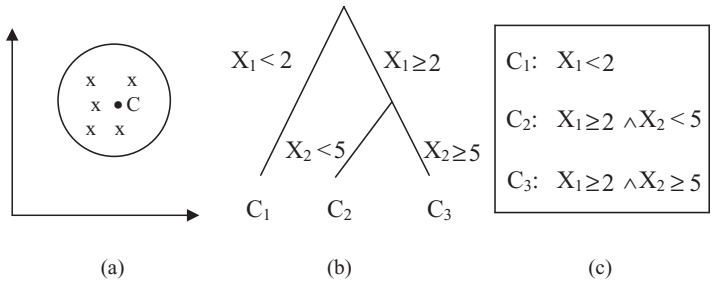


Figure 9.2. Different schemata for cluster representation. (a) Centroid; (b) clustering tree; (c) logical expressions.

separate set of points in X) and its descriptions or characterizations are generated as a result of a clustering procedure. There are several schemata for a formal description of discovered clusters:

1. Represent a cluster of points in an n-dimensional space (samples) by their centroid or by a set of distant (border) points in a cluster.
2. Represent a cluster graphically using nodes in a clustering tree.
3. Represent clusters by using logical expression on sample attributes.

Figure 9.2 illustrates these ideas. Using the centroid to represent a cluster is the most popular schema. It works well when the clusters are compact or isotropic. When the clusters are elongated or non-isotropic, however, this schema fails to represent them properly.

The availability of a vast collection of clustering algorithms in the literature and also in different software environments can easily confound a user attempting to select an approach suitable for the problem at hand. It is important to mention that there is no clustering technique that is universally applicable in uncovering the variety of structures present in multidimensional data sets. The user's understanding of the problem and the corresponding data types will be the best criteria in selecting the appropriate method. Most clustering algorithms are based on the following two popular approaches:

1. hierarchical clustering, and
2. iterative square-error partitional clustering.

Hierarchical techniques organize data in a nested sequence of groups, which can be displayed in the form of a dendrogram or a tree structure. Square-error partitional algorithms attempt to obtain the partition that minimizes the within-cluster scatter or maximizes the between-cluster scatter. These methods are nonhierarchical because all resulting clusters are groups of samples at the same level of partition. To guarantee that an optimum solution has been obtained, one has to examine all possible partitions of the N samples with n dimensions into K clusters (for a given K), but that retrieval

process is not computationally feasible. Notice that the number of all possible partitions of a set of N objects into K clusters is given by:

$$1/K! \sum_{j=1}^{K} \binom{K}{j} j^N$$

So various heuristics are used to reduce the search space, but then there is no guarantee that the optimal solution will be found.

Hierarchical methods that produce a nested series of partitions are explained in Section 6.3, while partitional methods that produce only one level of data grouping are given with more details in Section 9.4. The next section introduces different measures of similarity between samples; these measures are the core component of every clustering algorithm.

9.2 SIMILARITY MEASURES

To formalize the concept of a similarity measure, the following terms and notation are used throughout this chapter. A sample *x* (or feature vector, observation) is a single-data vector used by the clustering algorithm in a space of samples X. In many other texts, the term pattern is used. We do not use this term because of a collision in meaning with patterns as in pattern-association analysis, where the term has a totally different meaning. Most data samples for clustering take the form of finite dimensional vectors, and it is unnecessary to distinguish between an object or a sample x_i, and the corresponding vector. Accordingly, we assume that each sample $x_i \in X$, $i = 1, \ldots, n$ is represented by a vector $x_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$. The value m is the number of dimensions (features) of samples, while n is the total number of samples prepared for a clustering process that belongs to the sample domain X.

A sample can describe either a physical object (a chair) or an abstract object (a style of writing). Samples, represented conventionally as multidimensional vectors, have each dimension as a single feature. These features can be either quantitative or qualitative descriptions of the object. If the individual scalar component x_{ij} of a sample x_i is a feature or attribute value, then each component x_{ij}, $j = 1, \ldots, m$ is an element of a domain P_j, where P_j could belong to different types of data such as binary ($P_j = \{0,1\}$), integer ($P_j \subseteq Z$), real number ($P_j \subseteq R$), or a categorical set of symbols. In the last case, for example, Pj may be a set of colors: $P_j = \{$white, black, red, blue, green$\}$. If weight and color are two features used to describe samples, then the sample (20, black) is the representation of a black object with 20 units of weight. The first feature is quantitative and the second one is qualitative. In general, both feature types can be further subdivided, and details of this taxonomy are already given in Chapter 1.

Quantitative features can be subdivided as

1. *continuous values* (e.g., real numbers where $P_j \subseteq R$),
2. *discrete values* (e.g., binary numbers $P_j = \{0,1\}$, or integers $P_j \subseteq Z$), and
3. *interval values* (e.g., $P_j = \{x_{ij} \leq 20, 20 < x_{ij} < 40, x_{ij} \geq 40\}$.

Qualitative features can be

1. *nominal or unordered* (e.g., color is "blue" or "red"), and
2. *ordinal* (e.g., military rank with values "general" and "colonel").

Since similarity is fundamental to the definition of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering algorithms. This measure must be chosen very carefully because the quality of a clustering process depends on this decision. It is most common to calculate, instead of the similarity measure, the dissimilarity between two samples using a distance measure defined on the feature space. A distance measure may be a metric or a quasi-metric on the sample space, and it is used to quantify the dissimilarity of samples.

The word "similarity" in clustering means that the value of $s(x, x')$ is large when x and x' are two similar samples; the value of $s(x, x')$ is small when x and x' are not similar. Moreover, a similarity measure s is symmetric:

$$s(x, x') = s(x', x), \forall x, x' \in X$$

For most clustering techniques, we say that a similarity measure is normalized:

$$0 \leq s(x, x') \leq 1, \forall x, x' \in X$$

Very often a measure of dissimilarity is used instead of a similarity measure. A dissimilarity measure is denoted by $d(x, x')$, $\forall x, x' \in X$. Dissimilarity is frequently called a distance. A distance $d(x, x')$ is small when x and x' are similar; if x and x' are not similar $d(x, x')$ is large. We assume without loss of generality that

$$d(x, x') \geq 0, \forall x, x' \in X$$

Distance measure is also symmetric:

$$d(x, x') = d(x', x), \forall x, x' \in X$$

and if it is accepted as a *metric distance measure*, then a triangular inequality is required:

$$d(x, x'') \leq d(x, x') + d(x', x''), \forall x, x', x'' \in X$$

The most well-known metric distance measure is the Euclidean distance in an m-dimensional feature space:

$$d_2(x_i, x_j) = \left(\sum_{k=1}^{m} (x_{ik} - x_{jk})^2 \right)^{1/2}$$

Another metric that is frequently used is called the L_1 metric or city block distance:

$$d_1(x_i, x_j) = \sum_{k=1}^{m} |x_{ik} - x_{jk}|$$

and finally, the Minkowski metric includes the Euclidean distance and the city block distance as special cases:

$$d_p(x_i, x_j) = \left(\sum_{k=1}^{m} (x_{ik} - x_{jk})^p \right)^{1/p}$$

It is obvious that when p = 1, then d coincides with L_1 distance, and when p = 2, d is identical with the Euclidean metric. For example, for 4-D vectors $x_1 = \{1, 0, 1, 0\}$ and $x_2 = \{2, 1, -3, -1\}$, these distance measures are $d_1 = 1 + 1 + 4 + 1 = 7$, $d_2 = (1 + 1 + 16 + 1)^{1/2} = 4.36$, and $d_3 = (1 + 1 + 64 + 1)^{1/3} = 4.06$.

The Euclidian n-dimensional space model offers not only the Euclidean distance but also other measures of similarity. One of them is called the cosine-correlation:

$$s_{cos}(x_i, x_j) = \left[\sum_{k=1}^{m} (x_{ik} \cdot x_{jk}) \right] \Big/ \left[\sum_{k=1}^{m} x_{ik}^2 \sum_{k=1}^{m} x_{jk}^2 \right]^{1/2}$$

It is easy to see that

$$s_{cos}(x_i, x_j) = 1 \Leftrightarrow \forall i, j \text{ and } \lambda > 0 \text{ where } x_i = \lambda \cdot x_j$$

$$s_{cos}(x_i, x_j) = -1 \Leftrightarrow \forall i, j \text{ and } \lambda < 0 \text{ where } x_i = \lambda \cdot x_j$$

For the previously given vectors x_1 and x_2, the corresponding cosine measure of similarity is $s_{cos}(x_1, x_2) = (2 + 0 - 3 + 0)/(2^{½} \cdot 15^{½}) = -0.18$.

Computing distances or measures of similarity between samples that have some or all features that are noncontinuous is problematic, since the different types of features are not comparable and one standard measure is not applicable. In practice, different distance measures are used for different features of heterogeneous samples. Let us explain one possible distance measure for binary data. Assume that each sample is represented by the n-dimensional vector x_i, which has components with binary values ($v_{ij} \in \{0,1\}$). A conventional method for obtaining a distance measure between two samples x_i and x_j represented with binary features is to use the 2×2 contingency table for samples x_i and x_j, as shown in Table 9.2.

The meaning of the table parameters a, b, c, and d, which are given in Figure 6.2, is as follows:

1. *a* is the number of binary attributes of samples x_i and x_j such that $x_{ik} = x_{jk} = 1$.
2. *b* is the number of binary attributes of samples x_i and x_j such that $x_{ik} = 1$ and $x_{jk} = 0$.
3. *c* is the number of binary attributes of samples x_i and x_j such that $x_{ik} = 0$ and $x_{jk} = 1$.
4. *d* is the number of binary attributes of samples x_i and x_j such that $x_{ik} = x_{jk} = 0$.

TABLE 9.2. The 2×2 Contingency Table

		x_j	
x_i		1	0
	1	a	b
	0	c	d

For example, if x_i and x_j are 8-D vectors with binary feature values

$$x_i = \{0,0,1,1,0,1,0,1\}$$
$$x_j = \{0,1,1,0,0,1,0,0\}$$

then the values of the parameters introduced are

$$a = 2, b = 2, c = 1, \text{and } d = 3.$$

Several similarity measures for samples with binary features are proposed using the values in the 2×2 contingency table. Some of them are

1. simple matching coefficient (SMC)

$$s_{smc}(x_i, x_j) = (a + d)/(a + b + c + d)$$

2. Jaccard Coefficient

$$s_{jc}(x_i, x_j) = a/(a + b + c)$$

3. Rao's Coefficient

$$s_{rc}(x_i, x_j) = a/(a + b + c + d)$$

For the previously given 8-D samples x_i and x_j these measures of similarity will be $s_{smc}(x_i, x_j) = 5/8$, $s_{jc}(x_i, x_j) = 2/5$, and $s_{rc}(x_i, x_j) = 2/8$.

How to measure distances between values when categorical data are not binary? The simplest way to find similarity between two categorical attributes is to assign a similarity of 1 if the values are identical and a similarity of 0 if the values are not identical. For two multivariate categorical data points, the similarity between them will be directly proportional to the number of attributes in which they match. This simple measure is also known as the *overlap measure* in the literature. One obvious drawback of the overlap measure is that it does not distinguish between the different values taken by an attribute. All matches, as well as mismatches, are treated as equal.

This observation has motivated researchers to come up with data-driven similarity measures for categorical attributes. Such measures take into account the frequency

distribution of different attribute values in a given data set to define similarity between two categorical attribute values. Intuitively, the use of additional information would lead to a better performance. There are two main characteristics of categorical data that are included in new measures of similarity (distance):

1. number of values taken by each attribute, n_k (one attribute might take several hundred possible values, while another attribute might take very few values); and
2. distribution $f_k(x)$, which refers to the distribution of frequency of values taken by an attribute in the given data set.

Almost all similarity measures assign a similarity value between two d-dimensional samples X and Y belonging to the data set D as follows:

$$S(X,Y) = \sum_{k=1}^{d} w_k S_k(X_k, Y_k)$$

where $S_k(X_k, Y_k)$ is the per-attribute similarity between two values for the categorical attribute A_k. The quantity w_k denotes the weight assigned to the attribute A_k. To understand how different measures calculate the per-attribute similarity, $S_k(X_k; Y_k)$, consider a categorical attribute A, which takes one of the values{a, b, c, d}. The per-attribute similarity computation is equivalent to constructing the (symmetric) matrix shown in Table 9.3.

Essentially, in determining the similarity between two values, any categorical measure is filling the entries of this matrix. For example, the overlap measure sets the diagonal entries to 1 and the off-diagonal entries to 0, that is, the similarity is 1 if the values match and 0 if the values mismatch. Additionally, measures may use the following information in computing a similarity value (all the measures in this paper use only this information):

1. f(a), f(b), f(c), and f(d), the frequencies of the values in the data set;
2. N, the size of the data set; and
3. n, the number of values taken by the attribute (4 in the case above).

TABLE 9.3. Similarity Matrix for a Single Categorical Attribute

	a	b	c	d
a	S(a,a)	S(a,b)	S(a,c)	S(a,d)
b		S(b,b)	S(b,c)	S(b,d)
c			S(c,c)	S(c,d)
d				S(d,d)

TABLE 9.4. Goodall3 Similarity Measure for Categorical Attributes

Measure	$S_k(X_k, Y_k)$		w_k, k = 1, . . . , d
Goodall3	$1-p_k^2(X_k)$	if $X_k = Y_k$	1/d
	0	otherwise	

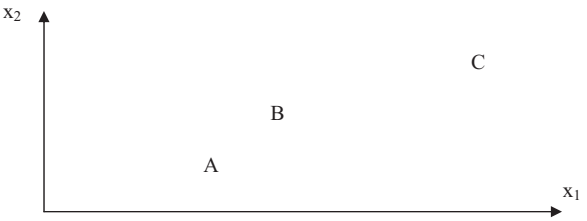


Figure 9.3. A and B are more similar than B and C using the MND measure.

We will present, as an illustrative example, only one additional measure of similarity for categorical data, *Goodall3*, because it shows good performances on average for a variety of experiments with different data sets. That does not mean that some other measures such as Eskin, Lin, Smirnov, or Burnaby will not be more appropriate for a specific data set. The *Goodall3 measure*, given in Table 9.4, assigns a high similarity if the matching values are infrequent regardless of the frequencies of the other values. The range of $S_k(X_k; Y_k)$ for matches in the *Goodall3* measure is $\left[0, 1-\frac{2}{N(N-1)}\right]$, with the minimum value being attained if X_k is the only value for attribute A_k and maximum value is attained if X_k occurs only twice.

There are some advanced distance measures applicable to categorical data, and also to numerical data, that take into account the effect of the surrounding or neighboring points in the n-dimensional spaces of samples. These surrounding points are called contexts. The similarity between two points, x_i and x_j, with the given context, is measured using the *mutual neighbor distance* (MND), which is defined as

$$\text{MND}(x_i, x_j) = \text{NN}(x_i, x_j) + \text{NN}(x_j, x_i)$$

where $\text{NN}(x_i, x_j)$ is the neighbor number of x_j with respect to x_i. If x_i is the closest point to x_j, then $\text{NN}(x_i, x_j)$ is equal to 1, if it is the second closest point, $\text{NN}(x_i, x_j)$ is equal to 2, and so on. Figures 9.3 and 9.4 give an example of the computation and basic characteristics of the MND measure.

Points in Figures 9.3 and 9.4, denoted by A, B, C, D, E, and F, are 2-D samples with features x_1 and x_2. In Figure 9.3, the nearest neighbor of A is B using Euclidian distance, and B's nearest neighbor is A. So,

$$\text{NN}(A, B) = \text{NN}(B, A) = 1 \Rightarrow \text{MND}(A, B) = 2$$

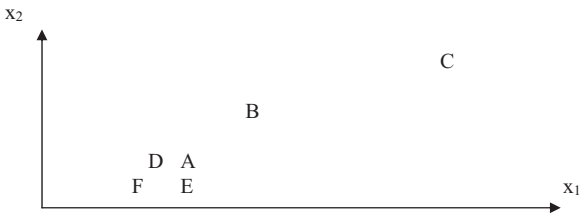


Figure 9.4. After changes in the context, B and C are more similar than A and B using the MND measure.

If we compute the distance between points B and C, the results will be

$$\text{NN(B, C)} = 1, \text{NN(C, B)} = 2 \Rightarrow \text{MND(B, C)} = 3$$

Figure 9.4 was obtained from Figure 9.3 by adding three new points D, E, and F (samples in the data set). Now, because the context has changed, the distances between the same points A, B, and C have also changed:

$$\text{NN(A, B)} = 1, \text{NN(B, A)} = 4 \Rightarrow \text{MND(A, B)} = 5$$

$$\text{NN(B, C)} = 1, \text{NN(C, B)} = 2 \Rightarrow \text{MND(B, C)} = 3$$

The MND between A and B has increased by introducing additional points close to A, even though A and B have not moved. B and C points become more similar than points A and B. The MND measure is not a metric because it does not satisfy the triangle inequality. Despite this, MND has been successfully applied in several real-world clustering tasks.

In general, based on a distance measure between samples, it is possible to define a distance measure between clusters (set of samples). These measures are an essential part in estimating the quality of a clustering process, and therefore they are part of clustering algorithms. The widely used measures for distance between clusters C_i and C_j are

1. $D_{min}(C_i, C_j) = \min |p_i - p_j|$, where $p_i \in C_i$ and $p_j \in C_j$;
2. $D_{mean}(C_i, C_j) = |m_i - m_j|$, where m_i and m_j are centriods of C_i and C_j;
3. $D_{avg}(C_i, C_j) = 1/(n_i\, n_j) \Sigma \Sigma |p_i - p_j|$, where $p_i \in C_i$ and $p_j \in C_j$, and n_i and n_j are the numbers of samples in clusters C_i and C_j; and
4. $D_{max}(C_i, C_j) = \max |p_i - p_j|$, where $p_i \in C_i$ and $p_j \in C_j$.

9.3 AGGLOMERATIVE HIERARCHICAL CLUSTERING

In hierarchical-cluster analysis, we do not specify the number of clusters as a part of the input. Namely, the input to a system is (X, s), where X is a set of samples, and s

is a measure of similarity. An output from a system is a hierarchy of clusters. Most procedures for hierarchical clustering are not based on the concept of optimization, and the goal is to find some approximate, suboptimal solutions, using iterations for improvement of partitions until convergence. Algorithms of hierarchical cluster analysis are divided into the two categories, divisible algorithms and agglomerative algorithms. A *divisible algorithm* starts from the entire set of samples X and divides it into a partition of subsets, then divides each subset into smaller sets, and so on. Thus, a divisible algorithm generates a sequence of partitions that is ordered from a coarser one to a finer one. An *agglomerative algorithm* first regards each object as an initial cluster. The clusters are merged into a coarser partition, and the merging process proceeds until the trivial partition is obtained: All objects are in one large cluster. This process of clustering is a bottom-up process, where partitions are from a finer one to a coarser one. In general, agglomerative algorithms are more frequently used in real-world applications than divisible methods, and therefore we will explain the agglomerative approach in greater detail.

Most agglomerative hierarchical clustering algorithms are variants of the *single-link* or *complete-link* algorithms. These two basic algorithms differ only in the way they characterize the similarity between a pair of clusters. In the single-link method, the distance between two clusters is the *minimum* of the distances between all pairs of samples drawn from the two clusters (one element from the first cluster, the other from the second). In the complete-link algorithm, the distance between two clusters is the *maximum* of all distances between all pairs drawn from the two clusters. A graphical illustration of these two distance measures is given in Figure 9.5.

In either case, two clusters are merged to form a larger cluster based on minimum-distance criteria. Although the single-link algorithm is computationally simpler, from a practical viewpoint it has been observed that the complete-link algorithm produces more useful hierarchies in most applications.

As explained earlier, the only difference between the single-link and complete-link approaches is in the distance computation. For both, the basic steps of the agglomerative clustering algorithm are the same. These steps are as follows:

1. Place each sample in its own cluster. Construct the list of intercluster distances for all distinct unordered pairs of samples, and sort this list in ascending order.

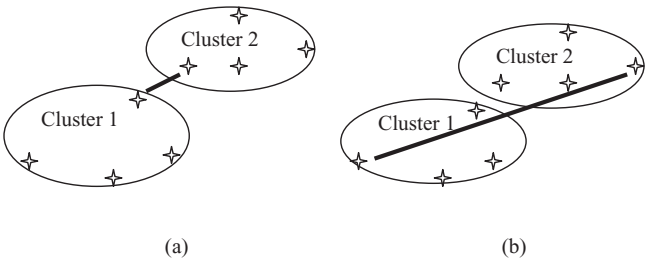


Figure 9.5. Distances for a single-link and a complete-link clustering algorithm. (a) Single-link distance; (b) complete-link distance.

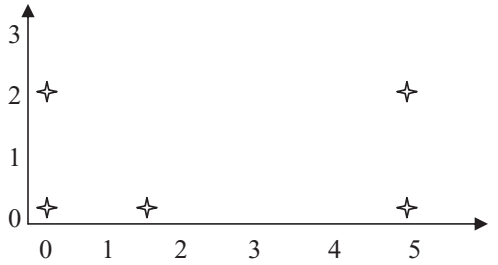


Figure 9.6. Five two-dimensional samples for clustering.

2. Step through the sorted list of distances, forming for each distinct threshold value d_k a graph of the samples where pairs of samples closer than d_k are connected into a new cluster by a graph edge. If all the samples are members of a connected graph, stop. Otherwise, repeat this step.
3. The output of the algorithm is a nested hierarchy of graphs, which can be cut at the desired dissimilarity level forming a partition (clusters) identified by simple connected components in the corresponding subgraph.

Let us consider five points $\{x_1, x_2, x_3, x_4, x_5\}$ with the following coordinates as a 2-D sample for clustering:

$$x_1 = (0, 2), x_2 = (0, 0), x_3 = (1.5, 0), x_4 = (5, 0), \text{and } x_5 = (5, 2).$$

For this example, we selected 2-D points because it is easier to graphically represent these points and to trace all the steps in the clustering algorithm. The points are represented graphically in Figure 9.6.

The distances between these points using the Euclidian measure are

$$d(x_1, x_2) = 2, d(x_1, x_3) = 2.5, d(x_1, x_4) = 5.39, d(x_1, x_5) = 5$$

$$d(x_2, x_3) = 1.5, d(x_2, x_4) = 5, d(x_2, x_5) = 5.29$$

$$d(x_3, x_4) = 3.5, d(x_3, x_5) = 4.03$$

$$d(x_4, x_5) = 2$$

The distances between points as clusters in the first iteration are the same for both single-link and complete-link clustering. Further computation for these two algorithms is different. Using agglomerative single-link clustering, the following steps are performed to create a cluster and to represent the cluster structure as a dendrogram.

First x_2 and x_3 samples are merged and a cluster $\{x_2, x_3\}$ is generated with a minimum distance equal to 1.5. Second, x_4 and x_5 are merged into a new cluster $\{x_4, x_5\}$ with a higher merging level of 2.0. At the same time, the minimum single-link distance between clusters $\{x_2, x_3\}$ and $\{x_1\}$ is also 2.0. So, these two clusters merge at the same level of similarity as x_4 and x_5. Finally, the two clusters $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$ are

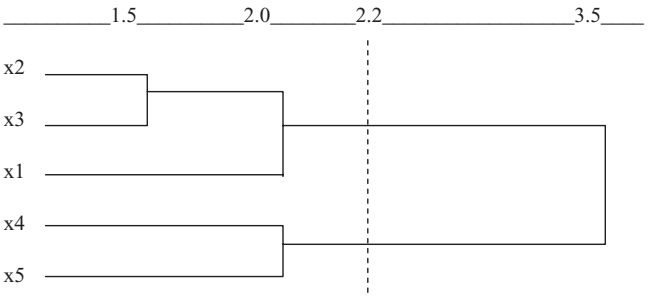


Figure 9.7. Dendrogram by single-link method for the data set in Figure 9.6.

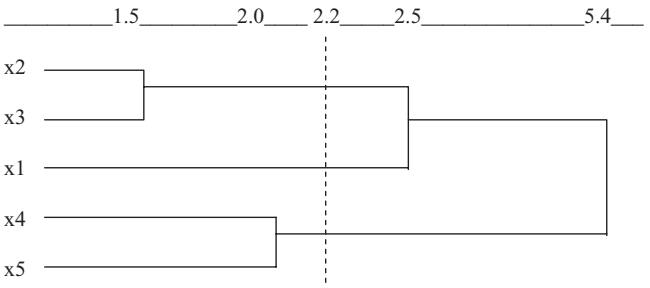


Figure 9.8. Dendrogram by complete-link method for the data set in Figure 9.6.

merged at the highest level with a minimum single-link distance of 3.5. The resulting dendrogram is shown in Figure 9.7.

The cluster hierarchy created by using an agglomerative complete-link clustering algorithm is different compared with the single-link solution. First, x_2 and x_3 are merged and a cluster $\{x_2, x_3\}$ is generated with the minimum distance equal to 1.5. Also, in the second step, x_4 and x_5 are merged into a new cluster $\{x_4, x_5\}$ with a higher merging level of 2.0. Minimal single-link distance is between clusters $\{x_2, x_3\}$, and $\{x_1\}$ is now 2.5, so these two clusters merge after the previous two steps. Finally, the two clusters $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$ are merged at the highest level with a minimal complete-link distance of 5.4. The resulting dendrogram is shown in Figure 9.8.

Selecting, for example, a threshold measure of similarity s = 2.2, we can recognize from the dendograms in Figures 9.7 and 9.8 that the final clusters for single-link and complete-link algorithms are not the same. A single-link algorithm creates only two clusters: $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$, while a complete-link algorithm creates three clusters: $\{x_1\}$, $\{x_2, x_3\}$, and $\{x_4, x_5\}$.

Unlike traditional agglomerative methods, *Chameleon* is a clustering algorithm that tries to improve the clustering quality by using a more elaborate criterion when merging two clusters. Two clusters will be merged if the interconnectivity and closeness of the merged clusters is very similar to the interconnectivity and closeness of the two individual clusters before merging.

To form the initial subclusters, Chameleon first creates a graph G = (V, E), where each node $v \in V$ represents a data sample, and a weighted edge $e(v_i, v_j)$ exists between two nodes v_i and v_j if v_j is one of the k-nearest neighbors of v_i. The weight of each edge in G represents the closeness between two samples, that is, an edge will weigh more if the two data samples are closer to each other. *Chameleon* then uses a graph-partition algorithm to recursively partition G into many small, unconnected subgraphs by doing a min-cut on G at each level of recursion. Here, a min-cut on a graph G refers to a partitioning of G into two parts of close, equal size such that the total weight of the edges being cut is minimized. Each subgraph is then treated as an initial subcluster, and the algorithm is repeated until a certain criterion is reached.

In the second phase, the algorithm goes bottom-up. *Chameleon* determines the similarity between each pair of elementary clusters C_i and C_j according to their relative interconnectivity $RI(C_i, C_j)$ and their relative closeness $RC(C_i, C_j)$. Given that the interconnectivity of a cluster is defined as the total weight of edges that are removed when a min-cut is performed, the relative interconnectivity $RI(C_i, C_j)$ is defined as the ratio between the interconnectivity of the merged cluster C_i and C_j to the average interconnectivity of C_i and C_j. Similarly, the relative closeness $RC(C_i, C_j)$ is defined as the ratio between the closeness of the merged cluster of C_i and C_j to the average internal closeness of C_i and C_j. Here the closeness of a cluster refers to the average weight of the edges that are removed when a min-cut is performed on the cluster.

The similarity function is then computed as a product: $RC(C_i, C_j) * RI(C_i, C_j)^{\alpha}$ where α is a parameter between 0 and 1. A value of 1 for α will give equal weight to both measures while decreasing α will place more emphasis on $RI(C_i, C_j)$. Chameleon can automatically adapt to the internal characteristics of the clusters and it is effective in discovering arbitrarily shaped clusters of varying density. However, the algorithm is not effective for high-dimensional data having $O(n^2)$ time complexity for n samples.

9.4 PARTITIONAL CLUSTERING

Every partitional-clustering algorithm obtains a single partition of the data instead of the clustering structure, such as a dendrogram, produced by a hierarchical technique. Partitional methods have the advantage in applications involving large data sets for which the construction of a dendrogram is computationally very complex. The partitional techniques usually produce clusters by optimizing a criterion function defined either locally (on a subset of samples) or globally (defined over all of the samples). Thus, we say that a clustering criterion can be either global or local. A global criterion, such as the Euclidean square-error measure, represents each cluster by a prototype or centroid and assigns the samples to clusters according to the most similar prototypes. A local criterion, such as the minimal MND, forms clusters by utilizing the local structure or context in the data. Therefore, identifying high-density regions in the data space is a basic criterion for forming clusters.

The most commonly used partitional-clustering strategy is based on the square-error criterion. The general objective is to obtain the partition that, for a fixed number of clusters, minimizes the total square-error. Suppose that the given set of N samples

in an n-dimensional space has somehow been partitioned into K clusters $\{C_1, C_2, \ldots, C_k\}$. Each C_k has n_k samples and each sample is in exactly one cluster, so that $\Sigma n_k = N$, where $k = 1, \ldots, K$. The mean vector M_k of cluster C_k is defined as the *centroid* of the cluster or

$$M_k = (1/n_k)\sum_{i=1}^{n_k} x_{ik}$$

where x_{ik} is the i^{th} sample belonging to cluster C_k. The square-error for cluster C_k is the sum of the squared Euclidean distances between each sample in C_k and its centroid. This error is also called the *within-cluster variation*:

$$e_k^2 = \sum_{i=1}^{n_k} (x_{ik} - M_k)^2$$

The square-error for the entire clustering space containing K clusters is the sum of the within-cluster variations:

$$E_k^2 = \sum_{k=1}^{K} e_k^2$$

The objective of a square-error clustering method is to find a partition containing K clusters that minimize E_k^2 for a given K.

The *K-means partitional-clustering algorithm* is the simplest and most commonly used algorithm employing a square-error criterion. It starts with a random, initial partition and keeps reassigning the samples to clusters, based on the similarity between samples and clusters, until a convergence criterion is met. Typically, this criterion is met when there is no reassignment of any sample from one cluster to another that will cause a decrease of the total squared error. K-means algorithm is popular because it is easy to implement, and its time and space complexity is relatively small. A major problem with this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function if the initial partition is not properly chosen.

The simple K-means partitional-clustering algorithm is computationally efficient and gives surprisingly good results if the clusters are compact, hyperspherical in shape, and well separated in the feature space. The basic steps of the K-means algorithm are

1. select an initial partition with K clusters containing randomly chosen samples, and compute the centroids of the clusters;
2. generate a new partition by assigning each sample to the closest cluster center;
3. compute new cluster centers as the centroids of the clusters; and
4. repeat steps 2 and 3 until an optimum value of the criterion function is found (or until the cluster membership stabilizes).

Let us analyze the steps of the K-means algorithm on the simple data set given in Figure 9.6. Suppose that the required number of clusters is two, and initially, clusters are formed from a random distribution of samples: $C_1 = \{x_1, x_2, x_4\}$ and $C_2 = \{x_3, x_5\}$. The centriods for these two clusters are

$$M_1 = \{(0+0+5)/3, (2+0+0)/3\} = \{1.66, 0.66\}$$

$$M_2 = \{(1.5+5)/2, (0+2)/2\} = \{3.25, 1.00\}$$

Within-cluster variations, after initial random distribution of samples, are

$$e_1^2 = [(0-1.66)^2 + (2-0.66)^2] + [(0-1.66)^2 + (0-0.66)^2] + [(5-1.66)^2 + (0-0.66)^2] = 19.36$$

$$e_2^2 = [(1.5-3.25)^2 + (0-1)^2] + [(5-3.25)^2 + (2-1)^2] = 8.12$$

And the total square-error is

$$E^2 = e_1^2 + e_2^2 = 19.36 + 8.12 = 27.48$$

When we reassign all samples, depending on a minimum distance from centroids M_1 and M_2, the new redistribution of samples inside clusters will be

$$d(M_1, x_1) = (1.66^2 + 1.34^2)^{1/2} = 2.14 \text{ and } d(M_2, x_1) = 3.40 \Rightarrow x_1 \in C_1$$

$$d(M_1, x_2) = 1.79 \text{ and } d(M_2, x_2) = 3.40 \Rightarrow x_2 \in C_1$$

$$d(M_1, x_3) = 0.83 \text{ and } d(M_2, x_3) = 2.01 \Rightarrow x_3 \in C_1$$

$$d(M_1, x_4) = 3.41 \text{ and } d(M_2, x_4) = 2.01 \Rightarrow x_4 \in C_2$$

$$d(M_1, x_5) = 3.60 \text{ and } d(M_2, x_5) = 2.01 \Rightarrow x_5 \in C_2$$

New clusters $C_1 = \{x_1, x_2, x_3\}$ and $C_2 = \{x_4, x_5\}$ have new centroids

$$M_1 = \{0.5, 0.67\}$$

$$M_2 = \{5.0, 1.0\}$$

The corresponding within-cluster variations and the total square-error are

$$e_1^2 = 4.17$$

$$e_2^2 = 2.00$$

$$E^2 = 6.17$$

We can see that after the first iteration, the total square-error is significantly reduced (from the value 27.48 to 6.17). In this simple example, the first iteration was at the same time the final one because if we analyze the distances between the new centroids and the samples, the latter will all be assigned to the same clusters. There is no reassignment and therefore the algorithm halts.

In summary, the K-means algorithm and its equivalent in an artificial neural networks domain—the Kohonen net—have been applied for clustering on large data sets. The reasons behind the popularity of the K-means algorithm are as follows:

1. Its time complexity is $O(n \times k \times l)$, where n is the number of samples, k is the number of clusters, and l is the number of iterations taken by the algorithm to converge. Typically, k and l are fixed in advance and so the algorithm has linear time complexity in the size of the data set.
2. Its space complexity is $O(k + n)$, and if it is possible to store all the data in the primary memory, access time to all elements is very fast and the algorithm is very efficient.
3. It is an order-independent algorithm. For a given initial distribution of clusters, it generates the same partition of the data at the end of the partitioning process irrespective of the order in which the samples are presented to the algorithm.

A big frustration in using iterative partitional-clustering programs is the lack of guidelines available for choosing K-number of clusters apart from the ambiguity about the best direction for initial partition, updating the partition, adjusting the number of clusters, and the stopping criterion. The K-means algorithm is very sensitive to noise and outlier data points, because a small number of such data can substantially influence the mean value. Unlike the K-means, the *K-mediods* method, instead of taking the mean value of the samples, uses the most centrally located object (mediods) in a cluster to be the cluster representative. Because of this, the K-mediods method is less sensitive to noise and outliers. *Fuzzy c-means*, proposed by Dunn and later improved, is an extension of K-means algorithm where each data point can be a member of multiple clusters with a membership value expressed through fuzzy sets. Despite its drawbacks, k-means remains the most widely used partitional clustering algorithm in practice. The algorithm is simple, easily understandable and reasonably scalable, and can be easily modified to deal with streaming data.

9.5 INCREMENTAL CLUSTERING

There are more and more applications where it is necessary to cluster a large collection of data. The definition of "large" has varied with changes in technology. In the 1960s, "large" meant several-thousand samples for clustering. Now, there are applications where millions of samples of high dimensionality have to be clustered. The algorithms discussed above work on large data sets, where it is possible to accommodate the entire data set in the main memory. However, there are applications where the entire data set cannot be stored in the main memory because of its size. There are currently three possible approaches to solve this problem:

1. The data set can be stored in a secondary memory and subsets of this data are clustered independently, followed by a merging step to yield a clustering of the entire set. We call this approach the divide-and-conquer approach.

2. An incremental-clustering algorithm can be employed. Here, data are stored in the secondary memory and data items are transferred to the main memory one at a time for clustering. Only the cluster representations are stored permanently in the main memory to alleviate space limitations.
3. A parallel implementation of a clustering algorithm may be used where the advantages of parallel computers increase the efficiency of the divide-and-conquer approach.

An incremental-clustering approach is most popular, and we will explain its basic principles. The following are the global steps of the incremental-clustering algorithm.

1. Assign the first data item to the first cluster.
2. Consider the next data item. Either assign this item to one of the existing clusters or assign it to a new cluster. This assignment is done based on some criterion, for example, the distance between the new item and the existing cluster centroids. In that case, after every addition of a new item to an existing cluster, recompute a new value for the centroid.
3. Repeat step 2 until all the data samples are clustered.

The space requirements of the incremental algorithm are very small, necessary only for the centroids of the clusters. Typically, these algorithms are non-iterative and therefore their time requirements are also small. But, even if we introduce iterations into the incremental-clustering algorithm, computational complexity and corresponding time requirements do not increase significantly. On the other hand, there is one obvious weakness of incremental algorithms that we have to be aware of. Most incremental algorithms do not satisfy one of the most important characteristics of a clustering process: order-independence. An algorithm is order-independent if it generates the same partition for any order in which the data set is presented. Incremental algorithms are very sensitive to the order of samples, and for different orders they generate totally different partitions.

Let us analyze the incremental-clustering algorithm with the sample set given in Figure 9.6. Suppose that the order of samples is x_1, x_2, x_3, x_4, x_5 and the threshold level of similarity between clusters is $\delta = 3$.

1. The first sample x_1 will become the first cluster $C_1 = \{x_1\}$. The coordinates of x_1 will be the coordinates of the centroid $M_1 = \{0, 2\}$.
2. Start analysis of the other samples.
   (a) Second sample x_2 is compared with M_1, and the distance d is determined

$$d(x_2, M_1) = (0^2 + 2^2)^{1/2} = 2.0 < 3$$

Therefore, x_2 belongs to the cluster C_1. The new centroid will be

$$M_1 = \{0, 1\}$$

(b) The third sample x_3 is compared with the centroid M_1 (still the only centroid):

$$d(x_3, M_1) = (1.5^2 + 1^2)^{1/2} = 1.8 < 3$$

$$x_3 \in C_1 \Rightarrow C_1 = \{x_1, x_2, x_3\} \Rightarrow M_1 = \{0.5,\ 0.66\}$$

(c) The fourth sample x_4 is compared with the centroid M_1:

$$d(x_4, M_1) = (4.5^2 + 0.66^2)^{1/2} = 4.55 > 3$$

Because the distance of the sample from the given centroid M_1 is larger than the threshold value δ, this sample will create its own cluster $C_2 = \{x_4\}$ with the corresponding centroid $M_2 = \{5, 0\}$.

(d) The fifth sample x_5 is compared with both cluster centroids:

$$d(x_5, M_1) = (4.5^2 + 1.44^2)^{1/2} = 4.72 > 3$$

$$d(x_5, M_2) = (0^2 + 2^2)^{1/2} = 2 < 3$$

The sample is closer to the centroid M_2, and its distance is less than the threshold value δ. Therefore, sample x_5 is added to the second cluster C_2:

$$C_2 = \{x_4, x_5\} \Rightarrow M_2 = \{5, 1\}$$

3. All samples are analyzed and a final clustering solution of two clusters is obtained:

$$C_1 = \{x_1, x_2, x_3\} \text{ and } C_2 = \{x_4, x_5\}$$

The reader may check that the result of the incremental-clustering process will not be the same if the order of the samples is different. Usually, this algorithm is not iterative (although it could be) and the clusters generated after all the samples have been analyzed in one iteration are the final clusters. If the iterative approach is used, the centroids of the clusters computed in the previous iteration are used as a basis for the partitioning of samples in the next iteration.

For most partitional-clustering algorithms, including the iterative approach, a summarized representation of the cluster is given through its clustering feature (CF) vector. This vector of parameters is given for every cluster as a triple, consisting of the number of points (samples) of the cluster, the centroid of the cluster, and the radius of the cluster. The cluster's radius is defined as the square-root of the average mean-squared distance from the centroid to the points in the cluster (averaged within-cluster variation). When a new point is added or removed from a cluster, the new CF can be computed from the old CF. It is very important that we do not need the set of points in the cluster to compute a new CF.

If samples are with categorical data, then we do not have a method to calculate centroids as representatives of the clusters. In that case, an additional algorithm called *K-nearest neighbor* may be used to estimate distances (or similarities) between samples and existing clusters. The basic steps of the algorithm are

1. to compute the distances between the new sample and all previous samples, already classified into clusters;
2. to sort the distances in increasing order and select K samples with the smallest distance values; and
3. to apply the voting principle. A new sample will be added (classified) to the largest cluster out of K selected samples.

For example, given six 6-D categorical samples

$$X_1 = \{A, B, A, B, C, B\}$$

$$X_2 = \{A, A, A, B, A, B\}$$

$$X_3 = \{B, B, A, B, A, B\}$$

$$X_4 = \{B, C, A, B, B, A\}$$

$$X_5 = \{B, A, B, A, C, A\}$$

$$X_6 = \{A, C, B, A, B, B\}$$

they are gathered into two clusters $C_1 = \{X_1, X_2, X_3\}$ and $C_2 = \{X_4, X_5, X_6\}$. How does one classify the new sample $Y = \{A, C, A, B, C, A\}$?

To apply the K-nearest neighbor algorithm, it is necessary, as the first step, to find all distances between the new sample and the other samples already clustered. Using the SMC measure, we can find similarities instead of distances between samples.

Similarities with Elements in C_1	Similarities with Elements in C_2
$SMC(Y, X_1) = 4/6 = 0.66$	$SMC(Y, X_4) = 4/6 = 0.66$
$SMC(Y, X_2) = 3/6 = 0.50$	$SMC(Y, X_5) = 2/6 = 0.33$
$SMC(Y, X_3) = 2/6 = 0.33$	$SMC(Y, X_6) = 2/6 = 0.33$

Using the 1-nearest neighbor rule ($K = 1$), the new sample cannot be classified because there are two samples (X_1 and X_4) with the same, highest similarity (smallest distances), and one of them is in the class C_1 and the other in the class C_2. On the other hand, using the 3-nearest neighbor rule ($K = 3$) and selecting the three largest similarities in the set, we can see that two samples (X_1 and X_2) belong to class C_1, and only one sample to class C_2. Therefore, using a simple voting system we can classify the new sample Y into the C_1 class.

9.6 DBSCAN ALGORITHM

Density-based approach in clustering assumes that clusters are regarded as dense regions of objects in the data space that are separated by regions of low object density (noise). These regions may have an arbitrary shape. Crucial concepts of this approach are density and connectivity both measured in terms of local distribution of nearest neighbors. The algorithm DBSCAN targeting low-dimensional data is the major representative in this category of density-based clustering algorithms. The main reason why DBSCAN recognizes the clusters is that within each cluster we have a typical density of points that is considerably higher than outside of the cluster. Furthermore, the points' density within the areas of noise is lower than the density in any of the clusters.

DBSCAN is based on two main concepts: *density reachability* and *density connectability*. These both concepts depend on two input parameters of the DBSCAN clustering: the size of epsilon neighborhood (ε) and the minimum points in a cluster (m). The key idea of the DBSCAN algorithm is that, for each point of a cluster, the neighborhood of a given radius ε has to contain at least a minimum number of points m, that is, the density in the neighborhood has to exceed some predefined threshold. For example, in Figure 9.9 point p has only two points in the neighborhood ε, while point q has eight. Obviously, the density around q is higher then around p.

Density reachability defines whether two close points belong to the same cluster. Point p_1 is density-reachable from p_2 if two conditions are satisfied: (1) the points are close enough to each other: distance $(p_1,p_2) < \varepsilon$, and (2) there are enough of points in ε neighborhood of p_2: distance$(r,p_2) > m$, where r are some database points. In the example represented in Figure 9.9, point p is reachable from point q. *Density connectivity* is the next building step of DBSCAN. Points p_0 and p_n are *density connected*, if there is a sequence of *density-reachable* points $(p_0, p_1, p_2, \ldots)$ from p_0 to p_n such that p_i+_1 is *density-reachable* from p_i. These ideas are translated into DBSCAN *cluster* as a set of all density connected points.

The clustering process is based on the classification of the points in the dataset as *core points*, *border points*, and *noise points* (examples are given in Fig. 9.10):

- A point is a *core point* if it has more than a specified number of points (m) within neighborhood ε. These are points that are at the interior of a cluster

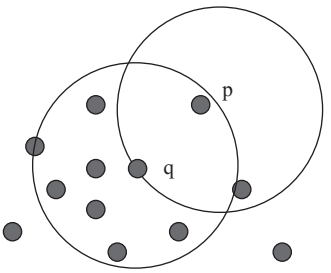


Figure 9.9. Neighborhood (ε) for points p and q.

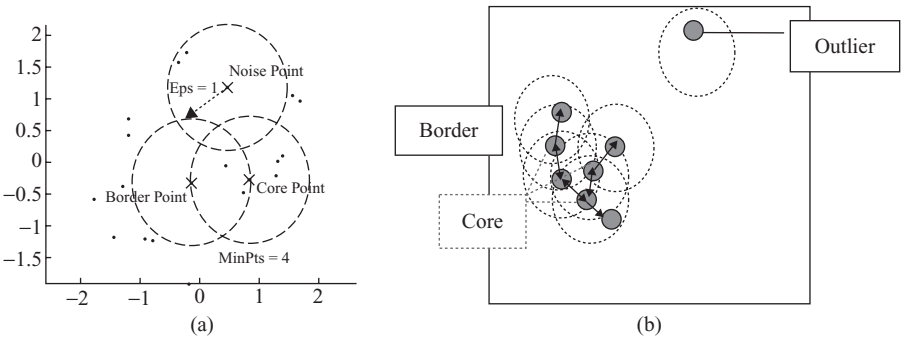


Figure 9.10. Examples of core, border, and noise points. (a) ε and m determine the type of the point; (b) core points build dense regions.

- A *border point* has fewer than m points within its neighborhood ε, but it is in the neighbor of a core point.
- A *noise point* is any point that is not a core point or a border point.

Ideally, we would have to know the appropriate parameters ε and m of each cluster. But there is no easy way to get this information in advance for all clusters of the database. Therefore, DBSCAN uses global values for ε and m, that is, the same values for all clusters. Also, numerous experiments indicate that DBSCAN clusters for $m > 4$ do not significantly differ from the case $m = 4$, while the algorithm needs considerably more computations. Therefore, in practice we may eliminate the parameter m by setting it to four for low-dimensional databases. The main steps of DBSCAN algorithm are as follows:

- Arbitrarily select a point $\boldsymbol{p}$.
- Retrieve all points density-reachable from $\boldsymbol{p}$ with respect to ε and m.
- If $\boldsymbol{p}$ is a core point, a new cluster is formed or existing cluster is extended.
- If $\boldsymbol{p}$ is a border point, no points are density-reachable from $\boldsymbol{p}$, and DBSCAN visits the next point of the database.
- Continue the process with other points in the database until all of the points have been processed.
- Since global values for ε and m are used, DBSCAN may merge two clusters into one cluster, if two clusters of different density are "close" to each other. The are close if the distance between clusters is lower than ε.

Examples of clusters obtained by DBSCAN algorithm are illustrated in Figure 9.11. Obviously, DBSCAN finds all clusters properly, independent of the size, shape, and location of clusters to each other.

The main advantages of the DBSCAN clustering algorithm are as follows:

1. DBSCAN does not require the number of clusters a priori, as opposed to K means and some other popular clustering algorithms.

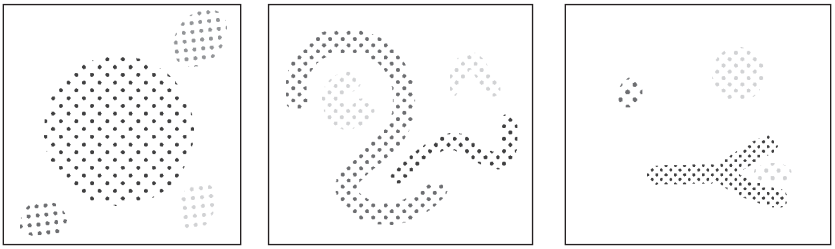


Figure 9.11. DBSCAN builds clusters of different shapes.

2. DBSCAN can find arbitrarily shaped clusters.
3. DBSCAN has a notion of noise and eliminate outliers from clusters.
4. DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database

DBSCAN also has some disadvantages. The complexity of the algorithm is still very high, although with some indexing structures it reaches O ($n \times \log n$). Finding neighbors is an operation based on distance, generally the Euclidean distance, and the algorithm may find the curse of dimensionality problem for high-dimensional data sets. Therefore, most applications of the algorithm are for low-dimensional real-world data.

9.7 BIRCH ALGORITHM

BIRCH is an efficient clustering technique for data in Euclidean vector spaces. The algorithm can efficiently cluster data with a single pass, and also it can deal effectively with outliers. BIRCH is based on the notion of a *CF* and a *CF tree*.

CF is a small representation of an underlying cluster that consists of one or many samples. BIRCH builds on the idea that samples that are close enough should always be considered as a group. CFs provide this level of abstraction with corresponding summarization of samples in a cluster. The idea is that a cluster of data samples can be represented by a triple of numbers (N, LS, SS), where N is the number of samples in the cluster, LS is the linear sum of the data points (vectors representing samples), and SS is the sum of squares of the data points. More formally, the component of vectors LS and SS are computed for every attribute X of data samples in a cluster:

$$\mathrm{LS}(X) = \sum_{i=1}^{N} X_i$$

$$\mathrm{SS}(X) = \sum_{i=1}^{N} X_i^2$$

In Figure 9.12 five 2-D samples are representing the cluster, and their CF summary is given with components: N = 5, LS = (16, 30), and SS = (54, 190). These are common statistical quantities, and a number of different cluster characteristics and intercluster

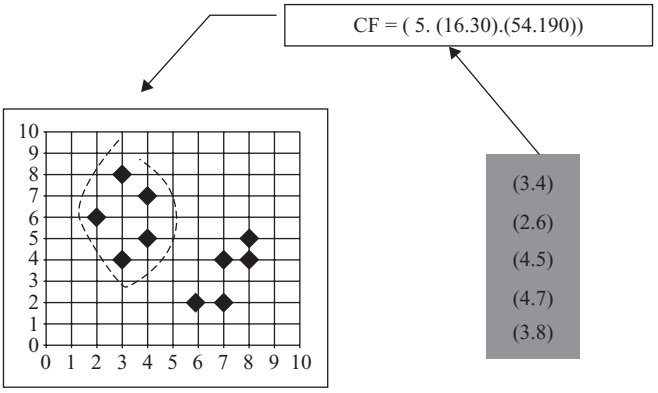


Figure 9.12. CF representation and visualization for a 2-D cluster.

distance measures can be derived from them. For example, we can compute the centroid for the cluster based on its CF representation, without revisiting original samples. Coordinates of the centroid are obtained by dividing the components of the LS vector by N. In our example the centroid will have the coordinates (3.2, 6.0). The reader may check on the graphical interpretation of the data (Fig. 9.12) that the position of the centroid is correct. The obtained summaries are then used instead of the original data for further clustering or manipulations with clusters. For example, if $CF_1 = (N_1, LS_1, SS_1)$ and $CF_2 = (N_2, LS_2, SS_2)$ are the CF entries of two disjoint clusters, then the CF entry of the cluster formed by merging the two clusters is

$$CF = CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

This simple equation shows us how simple the procedure is for merging clusters based on their simplified CF descriptions. That allows efficient incremental merging of clusters even for the streaming data.

BIRCH uses a hierarchical data structure called a CF tree for partitioning the incoming data points in an incremental and dynamic way. A CF tree is a height-balanced tree usually stored in a central memory. This allows fast lookups even when large data sets have been read. It is based on two parameters for nodes. CF nodes can have at maximum *B* children for non-leaf nodes, and a maximum of *L* entries for leaf nodes. Also, T is the threshold for the maximum diameter of an entry in the cluster. The CF tree size is a function of T. The bigger T is, the smaller the tree will be.

A CF tree (Fig 9.13) is built as the data sample is scanned. At every level of the tree a new data sample is inserted to the closest node. Upon reaching a leaf, the sample is inserted to the closest CF entry, as long as it is not overcrowded (diameter of the cluster D > T after the insert). Otherwise, a new CF entry is constructed and the sample is inserted. Finally, all CF statistics are updated for all nodes from the root to the leaf to represent the changes made to the tree. Since the maximum number of children per node (branching factor) is limited, one or several splits can happen. Building CF tree

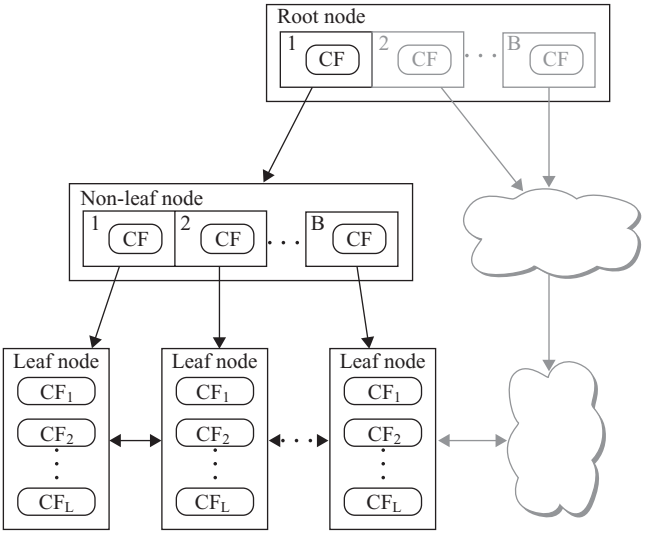


Figure 9.13. CF tree structure.

is only one, but the most important, phase in the BIRCH algorithm. In general, BIRCH employs four different phases during the clustering process:

1. Phase 1: *Scan all data and build an initial in-memory CF tree*
   It linearly scans all samples and inserts them in the CF tree as described earlier.
2. Phase 2: *Condense the tree to a desirable size by building a smaller CF tree*
   This can involve removing outliers and further merging of clusters.
3. Phase 3: *Global clustering*
   Employ a global clustering algorithm using the CF tree's leaves as input. CF features allow for effective clustering because the CF tree is very densely compressed in the central memory at this point. The fact that a CF tree is balanced allows the log-efficient search.
4. Phase 4: *Cluster refining*
   This is optional, and it requires more passes over the data to refine the results. All clusters are now stored in memory. If desired the actual data points can be associated with the generated clusters by reading all points from disk again.

BIRCH performs faster than most of the existing algorithms on large data sets. The algorithm can typically find a good clustering with a single scan of the data, and improve the quality further with a few additional scans (phases 3 and 4). Basic algorithm condenses metric data in the first pass using spherical summaries, and this part can be an incremental implementation. Additional passes cluster CFs to detect nonspherical clusters, and the algorithm approximates density function. There are several extensions of the algorithm that try to include nonmetric data, and that make applicability of the approach much wider.

9.8 CLUSTERING VALIDATION

How is the output of a clustering algorithm evaluated? What characterizes a "good" clustering result and a "poor" one? All clustering algorithms will, when presented with data, produce clusters regardless of whether the data contain clusters or not. Therefore, the first step in evaluation is actually an assessment of the data domain rather than the clustering algorithm itself. Data that we do not expect to form clusters should not be processed by any clustering algorithm. If the data do contain clusters, some clustering algorithms may obtain a "better" solution than others. Cluster validity is the second step, when we expect to have our data clusters. A clustering structure is valid if it cannot reasonably have occurred by chance or as an artifact of a clustering algorithm. Applying some of the available cluster methodologies, we assess the outputs. This analysis uses a specific criterion of optimality that usually contains knowledge about the application domain and therefore is subjective. There are three types of validation studies for clustering algorithms. An *external* assessment of validity compares the discovered structure with an a priori structure. An *internal* examination of validity tries to determine if the discovered structure is intrinsically appropriate for the data. Both assessments are subjective and domain-dependent. A *relative* test, as a third approach, compares the two structures obtained either from different cluster methodologies or by using the same methodology but with different clustering parameters, such as the order of input samples. This test measures their relative merit but we still need to resolve the question of selecting the structures for comparison.

Theory and practical applications both show that all approaches in the validation of clustering results have a subjective component. Hence, little in the way of "gold standards" exists in clustering evaluation. Recent studies in cluster analysis suggest that a user of a clustering algorithm should always keep the following issues in mind:

1. Every clustering algorithm will find clusters in a given data set whether they exist or not; the data should, therefore, be subjected to tests for clustering tendency before applying a clustering algorithm, followed by a validation of the clusters generated by the algorithm.
2. There is no best clustering algorithm; therefore, a user is advised to try several algorithms on a given data set.

It is important to remember that cluster analysis is an exploratory tool; the outputs of clustering algorithms only suggest or sometimes confirm hypotheses, but never prove any hypothesis about natural organization of data.

9.9 REVIEW QUESTIONS AND PROBLEMS

1. Why is the validation of a clustering process highly subjective?

2. What increases the complexity of clustering algorithms?

3. (a) Using MND distance, distribute the input samples given as 2-D points A(2, 2), B(4, 4), and C(7, 7) into two clusters.
(b) What will be the distribution of samples in the clusters if samples D(1, 1), E(2, 0), and F(0, 0) are added?

4. Given 5-D numeric samples A = (1, 0, 2, 5, 3) and B = (2, 1, 0, 3, −1), find
(a) the Eucledian distance between points;
(b) the city-block distance;
(c) the Minkowski distance for p = 3; and
(d) the cosine-correlation distance.

5. Given 6-D categorical samples C = (A, B, A, B, A, A) and D = (B, B, A, B, B, A), find
(a) an SMC of the similarity between samples;
(b) Jaccard's coefficient; and
(c) Rao's coefficient

6. Given a set of 5-D categorical samples

A = (1, 0, 1, 1, 0)
B = (1, 1, 0, 1, 0)
C = (0, 0, 1, 1, 0)
D = (0, 1, 0, 1, 0)
E = (1, 0, 1, 0, 1)
F = (0, 1, 1, 0, 0)

(a) Apply agglomerative hierarchical clustering using
(i) single-link similarity measure based on Rao's coefficient; and
(ii) complete-link similarity measure based on SMC.
(b) Plot the dendrograms for the solutions to parts (i) and (ii) of (a).

7. Given the samples X1 = {1, 0}, X2 = {0, 1}, X3 = {2, 1}, and X4 = {3, 3}, suppose that the samples are randomly clustered into two clusters C1 = {X1, X3} and C2 = {X2, X4}.
(a) Apply one iteration of the K-means partitional-clustering algorithm, and find a new distribution of samples in clusters. What are the new centroids? How can you prove that the new distribution of samples is better than the initial one?
(b) What is the change in the total square-error?
(c) Apply the second iteration of the K-means algorithm and discuss the changes in clusters.

8. For the samples in Problem 7, apply iterative clustering with the threshold value for cluster radius T = 2. What is the number of clusters and samples distribution after the first iteration?

9. Suppose that the samples in Problem 6 are distributed into two clusters:
C1 = {A, B, E} and C2 = {C, D, F}.

Using K-nearest neighbor algorithm, find the classification for the following samples:

(a) Y = {1 ,1 ,0 ,1 ,1} using K = 1
(b) Y = {1 ,1 ,0 ,1 ,1} using K = 3
(c) Z = {0, 1, 0, 0, 0} using K = 1
(d) Z = {0, 1, 0, 0, 0} using K = 5.

10. Implement the hierarchical agglomerative algorithm for samples with categorical values using the SMC measure of similarity.

11. Implement the partitional K-means clustering algorithm. Input samples are given in the form of a flat file.

12. Implement the incremental-clustering algorithm with iterations. Input samples are given in the form of a flat file.

13. Given the similarity matrix between five samples:

(a) Use the similarity matrix in the table to perform *complete link* hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged.
(b) How many clusters exist if the threshold similarity value is 0.5. Give the elements of each cluster.
(c) If DBSCAN algorithm is applied with threshold similarity of 0.6, and MinPts≥2 (required density), what are *core*, *border*,and *noise* points in the set of points p_i given in the table. Explain.

	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

14. Given the points x1 = {1, 0}, x2 = {0,1}, x3 = {2, 1}, and x4 = {3, 3}, suppose that these points are randomly clustered into two clusters: C1 = {x1, x3} and C2 = {x2, x4}. Apply one iteration of *K-means partitional clustering algorithm* and find new distribution of elements in clusters. What is the change in *total square-error*?

15. Answer True/False to the following statements. Discuss your answer if necessary.

(a) Running K-means with different initial seeds is likely to produce different results.
(b) Initial cluster centers have to be data points.
(c) Clustering stops when cluster centers are moved to the mean of clusters.
(d) K-means can be less sensitive to outliers if standard deviation is used instead of the average.
(e) K-means can be less sensitive to outliers if median is used instead of the average.

16. Identify the clusters in the Figure below using the center-, contiguity-, and density-based clustering. Assume center-based means *K-means,* contiguity-based means single link hierarchical and density-based means *DBSCAN*. Also indicate the

number of clusters for each case, and give a brief indication of your reasoning. Note that darkness or the number of dots indicates density.

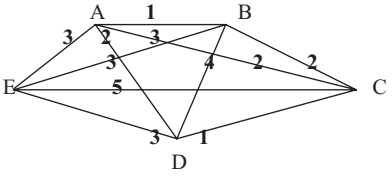
(a) (b) (c) (d)

17. Derive the mathematical relationship between cosine similarity and Euclidean distance when each data object has an L2 (Euclidean) length of 1.

18. Given a similarity measure with values in the interval [0, 1], describe two ways to transform this similarity value into a *dissimilarity* value in the interval [0, ∞].

19. Distances between samples (A, B, C, D, and E) are given in a graphical form:

Determine single-link and complete-link dendograms for the set of the samples.



20. There is a set S consisting of six points in the plane shown as below, a = (0, 0), b = (8, 0), c = (16, 0), d = (0, 6), e = (8, 6), f = (16, 6). Now we run the k-means algorithm on those points with k = 3. The algorithm uses the Euclidean distance metric (i.e., the straight line distance between two points) to assign each point to its nearest centroid. Also we define the following:

- *3-starting configuration* is a subset of three starting points from S that form the initial centroids, for example, {a, b, c}.
- *3-partition* is a partition of S into k nonempty subsets, for example, {a, b, e}, {c, d}, {f} is a 3-partition.

(a) How many 3-starting configurations are there?

(b) Fill in the last two columns of the following table.

3-partition	An example of a 3-starting configuration that can arrive at the 3-partition after 0 or more iterations of k-means	Number of unique 3-starting configurations
{a,b} {d,e} {c,f}		
{a} {d} {b, c, e, f}		
{a, b, d} {c} {e, f}		
{a, b} {d} {c, e, f}		

9.10 REFERENCES FOR FURTHER STUDY

Filippone, M., F. Camastra, F. Masulli, S. Rovetta, A Survey of Kernel and Spectral Methods for Clustering, *Pattern Recognition*, Vol. 41, 2008, pp. 176–190.

Clustering algorithms are a useful tool to explore data structures and have been employed in many disciplines. The focus of this paper is the partitioning clustering problem with a special interest in two recent approaches: kernel and spectral methods. The aim of this paper is to present a survey of kernel and spectral clustering methods, two approaches that are able to produce nonlinear separating hypersurfaces between clusters. The presented kernel clustering methods are the kernel version of many classical clustering algorithms, for example, *K*-means, SOM, and neural gas. Spectral clustering arises from concepts in spectral graph theory and the clustering problem is configured as a graph-cut problem where an appropriate objective function has to be optimized.

Han, J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Morgan Kaufmann, San Francisco, CA, 2006.

This book gives a sound understanding of data-mining principles. The primary orientation of the book is for database practitioners and professionals with emphasis on OLAP and data warehousing. In-depth analysis of association rules and clustering algorithms is the additional strength of the book. All algorithms are presented in easily understood pseudo-code and they are suitable for use in real-world, large-scale data-mining projects including advanced applications such as Web mining and text mining.

Hand, D., H. Mannila, P. Smith, *Principles of Data Mining*, MIT Press, Cambridge, MA, 2001.

The book consists of three sections. The first, foundations, provides a tutorial overview of the principles underlying data-mining algorithms and their applications. The second section, data-mining algorithms, shows how algorithms are constructed to solve specific problems in a principled manner. The third section shows how all of the preceding analyses fit together when applied to real-world data-mining problems.

Jain, A. K., M. N. Murty, P. J. Flynn, Data Clustering: A Review, *ACM Computing Surveys*, Vol. 31, No. 3, September 1999, pp. 264–323.

Although there are several excellent books on clustering algorithms, this review paper will give the reader enough details about the state-of-the-art techniques in data clustering, with an emphasis on large data sets problems. The paper presents the taxonomy of clustering techniques and identifies crosscutting themes, recent advances, and some important applications. For readers interested in practical implementation of some clustering methods, the paper offers useful advice and a large spectrum of references.

Miyamoto, S., *Fuzzy Sets in Information Retrieval and Cluster Analysis*, Cluver Academic Publishers, Dodrecht, Germany, 1990.

This book offers an in-depth presentation and analysis of some clustering algorithms and reviews the possibilities of combining these techniques with fuzzy representation of data. Information retrieval, which, with the development of advanced Web-mining techniques, is becoming more important in the data-mining community, is also explained in the book.