CHAPTER 7

Cluster Analysis

This chapter describes clustering, a process that classifies data without an existing or predefined model or output classes. It has a wide range of applications in web mining, bioinformatics, and so on.

7.1 INTRODUCTION

Cluster analysis is defined as the unsupervised classification of data into various clusters. We can also look upon cluster analysis as a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics. Clustering can be looked upon as a method of unsupervised learning under various machine-learning techniques. The application of cluster analysis can be found in numerous fields, namely, machine learning, pattern recognition, data mining, market research, social network analysis, image segmentation, and bioinformatics. Cluster analysis is an iterative process of knowledge discovery, which is closely related to other similar techniques such as automatic classification and numerical taxonomy.

Different kinds of application areas exist for clustering, and hence it is hard to find a unified clustering approach in general. Ideas, approaches, and heuristics developed in a certain field cannot be easily migrated or applied to another field. Hence, we look at the various fields and the similarities between them. Clustering is an unsupervised learning technique that groups data objects into a set of disjoint classes, called clusters, so that objects within a class show high similarity to each other, while objects in separate classes are more dissimilar, i.e., intraclass similarity is more and interclass similarity is less. Unsupervised means

that clustering does not rely on predefined model and output classes, while classifying data objects and classification refer to a procedure that assigns data objects to a set of classes.

Note: Clustering is an unsupervised learning technique in which a predefined model does not exist. Also there are no predefined output classes to which objects can be classified. Instead, based on clustering algorithms, data are grouped together and put to various classes based on a similarity measure.

7.2 CONCEPTS

Clustering is the process of organizing data into meaningful groups called clusters. It is not a new concept in computer science; it has existed since long back as classification and taxonomy. In selecting the clustering algorithm and for better clustering, knowledge about the type and source of the data is very useful. This type of clustering has found use in fields such as content mining and generally aims to give tags to the clusters.

In classification, we have information about the objects, characteristics we are searching for, and the available classifications, and it is more similar to just knowing where to place the new object in. Clustering, on the other hand, analyzes the data and finds its characteristics, either supervised or unsupervised [1, 2]. Also it reduces the number of bits required to convey information about a member such that much less information is required and extra information does not cause confusion. In this aspect, clustering is a form of data abstraction. The most general definition is that given N items, we can divide the N items into k groups based on the measure of similarity between the items, such that items in a group can be called "similar."

The following example (Fig. 7.1) demonstrates the clustering of balls of same type. There are nine balls, which are of three different types. We are interested in clustering the three different types of balls into three different groups.

Fig. 7.1

The balls of same types are clustered into groups, as shown in Fig. 7.2.

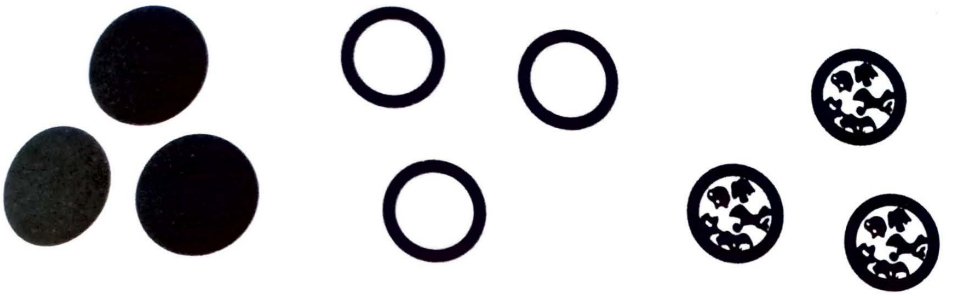

Fig. 7.2 Balls clustered into different groups

In clustering techniques, there is no prediction of classes, but the instances are divided into natural groups. These identified clusters presumably reflect some mechanism that causes certain cases to bear a strong resemblance to each other, and the technique helps in finding structure in the data.

Clustering methods can be hierarchy-based, density-based, grid-based, and model-based. Partitioning is another method of clustering in which various partitions are constructed and then evaluated by some criteria. In the hierarchical method, a hierarchical decomposition of the set of data is created using some criterion. The density-based method is based on connectivity and density functions, and the grid-based method focuses on multiple-level granularity structure. The model-based approach develops models for each of the clusters with the aim of finding the best fit of that model to each other.

7.3 *K*-MEANS CLUSTERING

k-means clustering is the most commonly used partitioning method. It is the simplest unsupervised learning algorithm used for solving clustering problems. Assuming *k* clusters, the given data set is classified into a certain number of clusters using the algorithm procedure [3, 4]. The main aim is to define *k* centroids, one for each cluster. The description of the algorithm is as follows:

7.3.1 Basic *K*-Means Algorithm for Finding *K* Clusters

- Initially, the number of clusters must be known, or chosen to be *k* say.
- The initial step is to choose a set of *k* instances as centers of the clusters. The points are often chosen such that they are mutually "farthest apart" in some way.
- Next, the algorithm considers each instance and assigns it to the closest cluster.

- The cluster centroids are recalculated, either after each instance assignment, or after the whole cycle of re-assignments.
- This process is iterated. Figure 7.3 gives the flow chart for *k*-means algorithm.

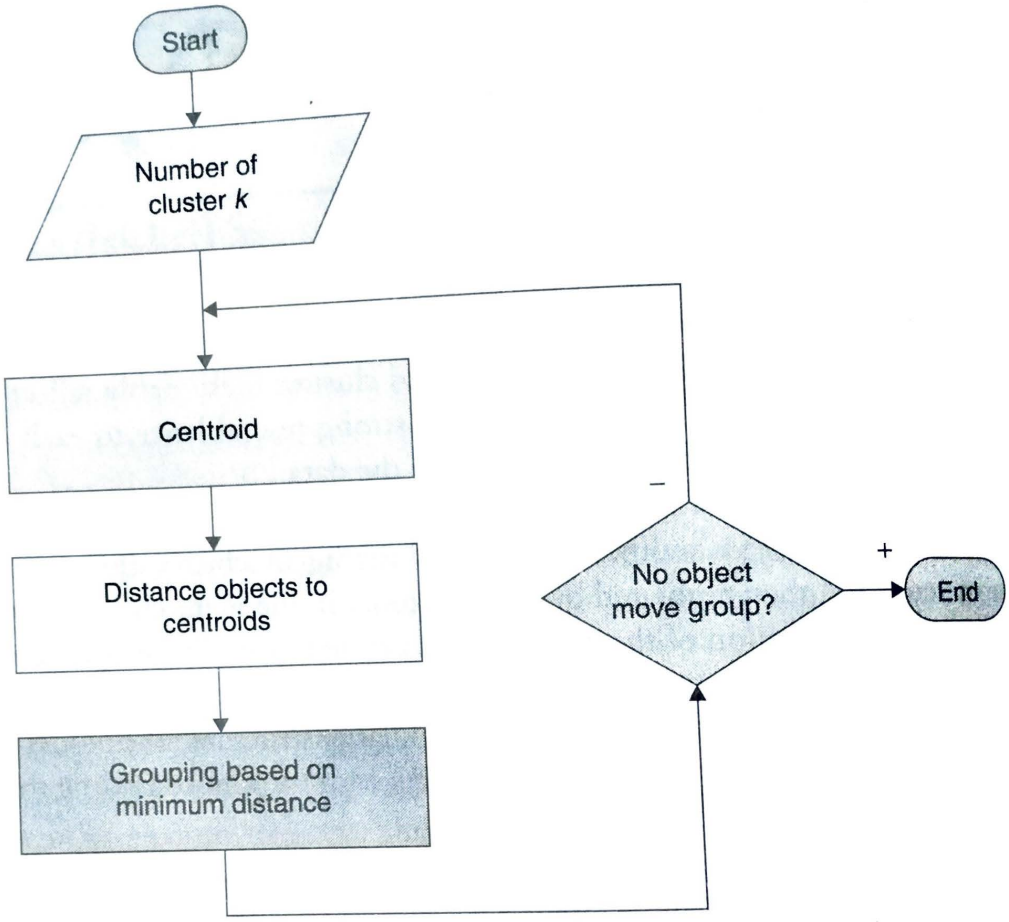

Fig. 7.3 Flow chart for *k*-means algorithm

Example Problem

Cluster the following seven points [with (u, v) representing locations] into three clusters: $P1(2, 10)$, $P2(2, 5)$, $P3(8, 4)$, $P4(5, 8)$, $P5(7, 5)$, $P6(6, 4)$, $P7(1, 2)$. The initial cluster centers are $P1(2, 10)$, $P4(5, 8)$, and $P7(1, 2)$. There can be different types of possible distance functions for clustering applications [5, 6].

Here the distance function used between two points $a = (u1, v1)$ and $b = (u2, v2)$ is defined as $\rho(a, b) = |u2 - u1| + |v2 - v1|$. The distance function can also be the Euclidian distance, $\text{Sqrt}[(u2 - u1)^2 + (v2 - v1)^2]$.

Use *k*-means algorithm in Fig. 7.3 to find the three cluster centers after the second iteration.

Solution

Iteration 1

First, all points in the first column of Table 7.1 are listed. The initial cluster centers—means (2, 10), (5, 8), and (1, 2)—are chosen randomly. By using the distance function, we will calculate the distance from the first point (2, 10) to each of the three means, in the next step.

Point	mean1
$u1, v1$	$u2, v2$
(2, 10)	**(2, 10)**

$$\rho(a, b) = |u2 - u1| + |v2 - v1|$$

$$\begin{aligned}\rho(\text{point, mean1}) &= |u2 - u1| + |v2 - v1| \\ &= |\mathbf{2 - 2}| + |\mathbf{10 - 10}| \\ &= 0 + 0 \\ &= 0\end{aligned}$$

Table 7.1 Progress of *k*-means algorithm (initial table)

	(2, 10)	(5, 8)	(1, 2)	
Point	**Dist. mean 1**	**Dist. mean 2**	**Dist. mean 3**	**Cluster**
P1 (2, 10)				
P2 (2, 5)				
P3 (8, 4)				
P4 (5, 8)				
P5 (7, 5)				
P6 (6, 4)				
P7 (1, 2)				

Point	mean2
$u1, v1$	$u2, v2$
(2, 10)	**(5, 8)**

$$\rho(a, b) = |u2 - u1| + |v2 - v1|$$

$$\rho(\text{point, mean2}) = |u2 - u1| + |v2 - v1|$$
$$= |\mathbf{5} - \mathbf{2}| + |\mathbf{8} - \mathbf{10}|$$
$$= 3 + 2$$
$$= 5$$

Similarly, distance with (1, 2),

$$\rho(a, b) = |u2 - u1| + |v2 - v1|$$
$$\rho(\text{point, mean2}) = |u2 - u1| + |v2 - v1|$$
$$= |\mathbf{1} - \mathbf{2}| + |\mathbf{2} - \mathbf{10}|$$
$$= 1 + 8$$
$$= 9$$

Assignment of $P1$ to cluster 1 is shown in Table 7.2.

Table 7.2 Progress of *k*-means algorithm (assignment of $P1$ to a cluster)

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist. mean 1**	**Dist. mean 2**	**Dist. mean 3**	**Cluster**
$P1$	(2, 10)	0	5	9	1
$P2$	(2, 5)				
$P3$	(8, 4)				
$P4$	(5, 8)				
$P5$	(7, 5)				
$P6$	(6, 4)				
$P7$	(1, 2)				

So to which cluster can we place the point (2, 10)? We can place the point (2, 10) to a cluster where the point has the shortest distance from the mean, that is, mean 1 (cluster 1), since the distance is 0.

Again, we go to the second point (2, 5). The distance to each of the three means is calculated by using the distance function.

Point $u1, v1$	mean1 $u2, v2$
(2, 5)	**(2, 10)**

$$\rho(a, b) = |u2 - u1| + |v2 - v1|$$

$$\rho(\text{point, mean1}) = |u2 - u1| + |v2 - v1|$$

$$= |\mathbf{2} - \mathbf{2}| + |\mathbf{10} - \mathbf{5}|$$

$$= 0 + 5$$

$$= 5$$

Point $u1, v1$	mean2 $u2, v2$
(2, 5)	**(5, 8)**

$$\rho(a, b) = |u2 - u1| + |v2 - v1|$$

$$\rho(\text{point, mean2}) = |u2 - u1| + |v2 - v1|$$

$$= |\mathbf{5} - \mathbf{2}| + |\mathbf{8} - \mathbf{5}|$$

$$= 3 + 3$$

$$= 6$$

Point $u1, v1$	mean3 $u2, v2$
(2, 5)	**(1, 8)**

$$\rho(a, b) = |u2 - u1| + |v2 - v1|$$

$$\rho(\text{point, mean2}) = |u2 - u1| + |v2 - v1|$$

$$= |\mathbf{1} - \mathbf{2}| + |\mathbf{2} - \mathbf{5}|$$

$$= 1 + 3$$

$$= 4$$

Now the values are filled in Table 7.3.

able 7.3 Progress of *k*-means algorithm (assigning *P*2 to a cluster)

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist. mean 1**	**Dist. mean 2**	**Dist. mean 3**	**Cluster**
*P*1	(2, 10)	0	5	9	1
*P*2	(2, 5)	5	6	4	3
*P*3	(8, 4)				
*P*4	(5, 8)				
*P*5	(7, 5)				
*P*6	(6, 4)				
*P*7	(1, 2)				

So in which cluster should the point (2, 5) be placed? The one where the point has the shortest distance from the mean, that is, mean 3 (cluster 3), since the distance is 0. It is shown in Table 7.3. Likewise, we fill in the rest of the table and place each point in one of the clusters.

Likewise we fill in the rest of the table and place each point in one of the clusters as shown in Table 7.4.

Table 7.4 Progress of *k*-means algorithm (end of first iteration)

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist. mean 1**	**Dist. mean 2**	**Dist. mean 3**	**Cluster**
*P*1	(2, 10)	0	5	9	1
*P*2	(2, 5)	5	6	4	3
*P*3	(8, 4)	12	7	9	2
*P*4	(5, 8)	5	0	10	2
*P*5	(7, 5)	10	5	9	2
*P*6	(6, 4)	10	5	7	2
*P*7	(1, 2)	9	10	0	3

Cluster 1	Cluster 2	Cluster 3
(2, 10)	(8, 4)	(2, 5)
	(5, 8)	(1, 2)
	7, 5)	
	(6, 4)	

Next we need to re-compute the new cluster centers. For this, take the mean of all points in each cluster. For cluster 1, we only have one point $p1(2, 10)$, which was the old mean; so the cluster center remains the same.

For cluster 2, we have $\left[\frac{(8+5+7+6)}{5}, \frac{(4+8+5+4)}{5}\right] = (5, 4)$.

For cluster 3, we have $\left[\frac{(2+1)}{2}, \frac{(5+2)}{2}\right] = (1.5, 3.5)$.

That was iteration 1. Next we go to iteration 2, iteration 3, and so on until there are no more changes in mean.

In iteration 2, we basically repeat the process explained in iteration 1, this time using the new means we computed.

7.3.2 Time and Space Complexity

The time complexity for k clusters is $O(I^*K^*m^*n)$, where I is the number of iterations required for convergence, m is the number of points, and n is the number of attributes. Since only the vectors are stored, the space requirements are basically $O(mn)$. I is typically small, and since most changes occur in the first few iterations, it can also be easily bounded. k-means is linear in m, the number of points, and is simple and efficient, provided the number of clusters is significantly less than m.

7.3.3 Updating Centroids Incrementally

The centroids can be updated incrementally, as each point is assigned to a cluster. Also adjustments may be made in the relative weight of the point being added. The goal is to achieve better accuracy and faster convergence. However, it may be difficult to make a good choice for the relative weight. Another advantage of incremental update is that empty clusters are not produced. Centroid updates are performed only after all points have been assigned to clusters. Only then we observe empty clusters.

Example

Assume that we have n sample feature vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, all from the same class, and they fall into k compact clusters ($k < n$). Let m_i be the mean of the vectors in cluster i. We can use a minimum-distance classifier to separate

them, provided the clusters are well separated. That is, we can say that v is in cluster i if $\| \mathbf{v} - \mathbf{m}_i \|$ is the minimum of all the k distances. This suggests the following procedure for finding the k-means:

1. Make initial guesses for the means $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k$
2. Until there are no changes in any mean,
   a. Use the estimated means to classify the samples into clusters
   b. For i from 1 to k
   c. Replace $\mathbf{m}_i$ with the mean of all of the samples for cluster i
   d. End for
3. End until

Here is an example (Fig. 7.4) showing how the means $\mathbf{m}_1$ and $\mathbf{m}_2$ move into the centers of two clusters.

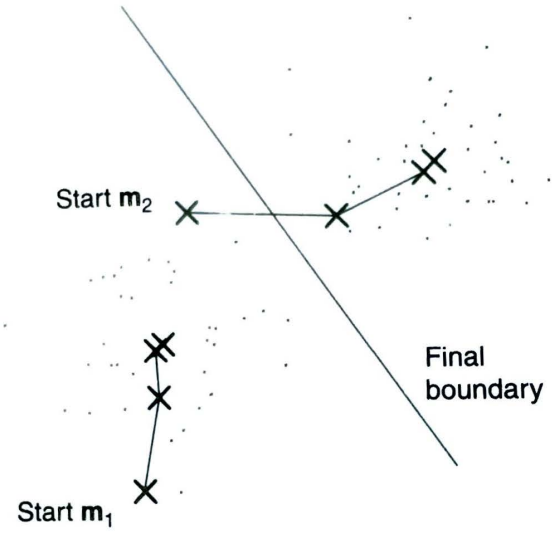

Fig. 7.4 Movement of means in clusters

REMARKS We have seen a simple version of k-means procedure that can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. Some of its weaknesses are as follows:

1. The way to initialize the means was not specified. One way to start is to randomly choose k of the samples.
2. For means, the results produced depend on the initial values, and suboptimal partitions are found frequently. The standard solution is to try a number of different starting points.
3. The results depend on the metric used to measure $\| \mathbf{v} - \mathbf{m}_i \|$. A popular solution is to normalize each variable by its standard deviation. The results also depend on the value of k.

7.4 DENSITY-BASED CLUSTERING

Density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering algorithm that works with a number of different distance metrics. When DBSCAN has processed a set of data points, a point will either be in a cluster or will be classified as noise. DBSCAN is based on the concepts of a point being "density-reachable" and "density-connected." Conventional clustering algorithms form clusters of spherical shapes and so on. But DBSCAN can form clusters of any arbitrary shape like "*S*" or oval shapes as in Fig. 7.5.

DBSCAN is developed based on the notions of "density" and "attraction" of data objects. The main idea is to consider a cluster as a high-dimensional dense area, where data objects are "attracted" with each other. The objects are packed closely with each other at the core part of the dense area and thereby have high density. Objects at the peripheral area of the cluster are attracted to the core part of the dense area and relatively sparsely distributed. Conceptually, data points fall into three classes:

Core points: These points are at the interior of a cluster. An interior point has enough points in its neighborhood. If two core points belong to each other's neighborhoods, then the core points belong to the same cluster.

Border points: A border point is a point that is not a core point, i.e., there are not enough points in its neighborhood, but it falls within the neighborhood of a core point. A border point may fall within the neighborhoods of core points from several different clusters.

Noise points: A noise point is any point that is not a core point or a border point.

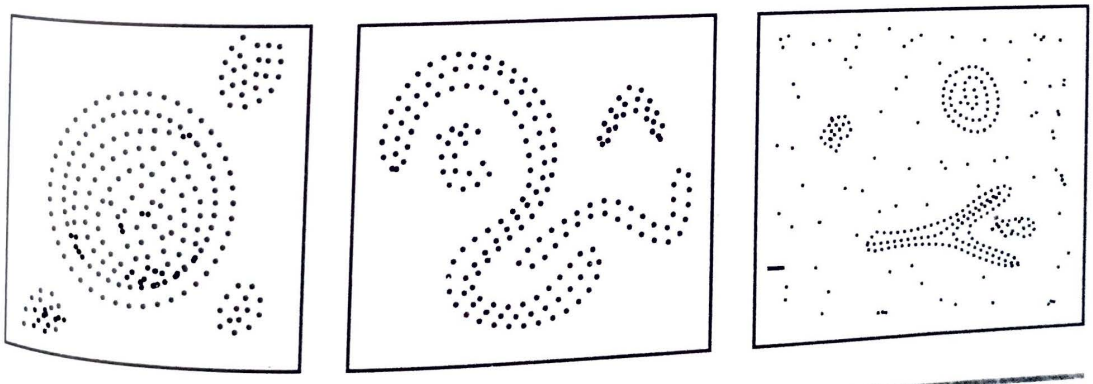

Fig. 7.5 Density concepts in DBSCAN

7.4.1 Density Concepts

Two global parameters:

Eps: Maximum radius of the neighborhood.

MinPts: Minimum number of points in an *Eps*-neighborhood of that point.

Object: Object with at least *MinPts* objects within a radius "*Eps*-neighborhood."
Border object: Object on the border of a cluster.

7.4.2 Density-Based Clustering: Background

Density-reachable: A point p is density-reachable from a point q with respect to *Eps* and *MinPts* if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Density-connected: A point p is density-connected to a point q with respect to *Eps* and *MinPts* if there is a point o such that both p and q are density-reachable from o with respect to *Eps* and *MinPts*.

7.4.3 DBSCAN: The Algorithm

1. Arbitrarily select a point p.
2. Retrieve all points density-reachable from p with respect to *Eps* and *MinPts*.
3. If p is a core point, a cluster is formed.
4. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database.
5. Continue the process until all the points have been processed.

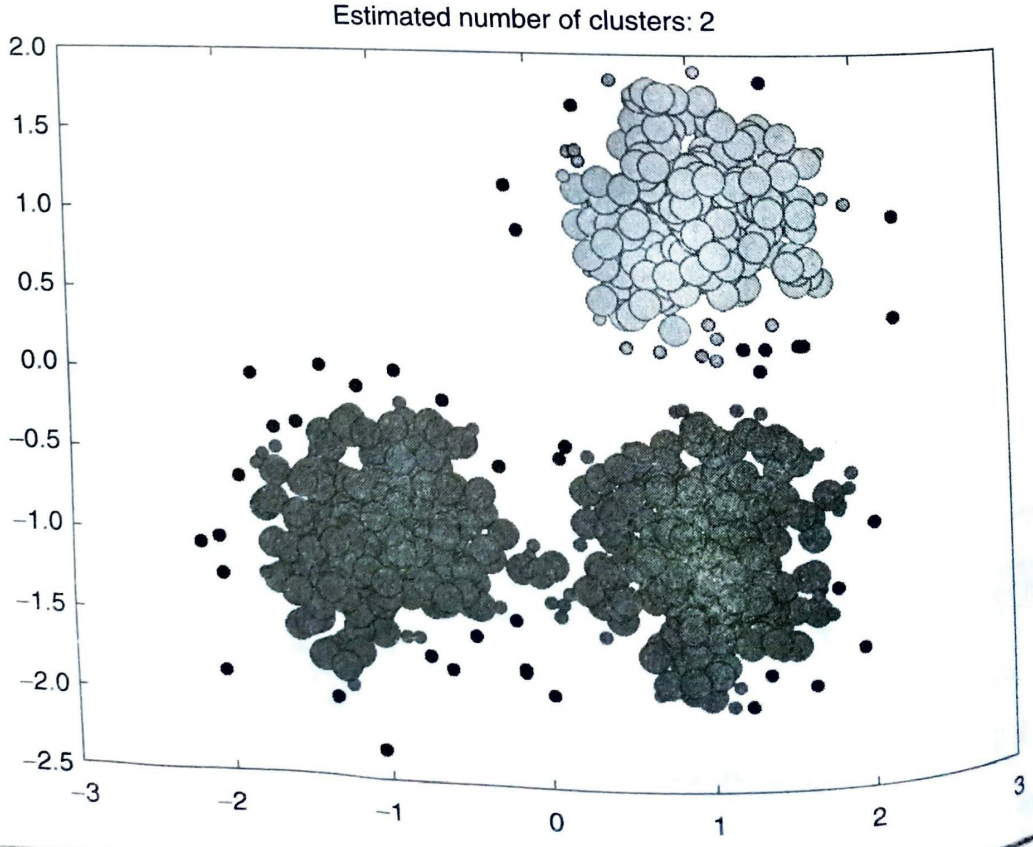

Fig. 7.6 A DBSCAN clustering result

Example

Consider Fig. 7.7 for a given *Eps*, represented by the radius of the circles, and let *MinPts* = 3. Of the labeled points, *m*, *p*, *o*, and *r* are core objects because each of them is in an *Eps*-neighborhood containing at least three points. Object *q* is directly density-reachable from *m*. Object *m* is directly density-reachable from *p* and vice versa.

Object *q* is directly density-reachable from *m*, and *m* is directly density-reachable from *p*, hence object *q* is (indirectly) density-reachable from *p*. Because *q* is not a core object, *p* is not density-reachable from *q*. Similarly *r* and *s* are density-reachable from *o*, and *o* is density-reachable from *r*. Thus, *o*, *r*, and *s* are all density-connected. The closure of density-connectedness can be used to find connected dense regions as clusters.

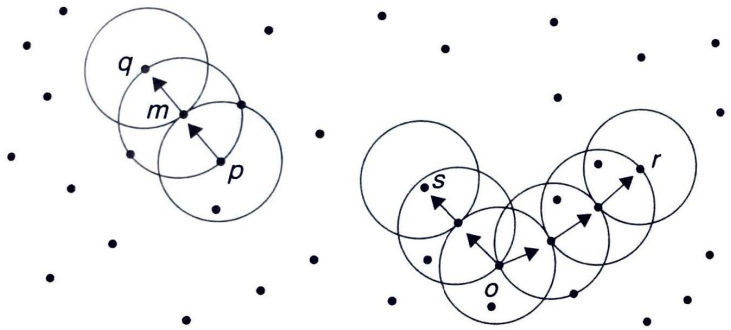

Fig. 7.7 Density-reachability and density-connectivity

In Fig. 7.6, color indicates cluster membership, with large circles indicating core samples found by the algorithm. Smaller circles are noncore samples that are still part of a cluster. Moreover, the outliers are indicated by black points below.

7.4 WEIGHTED GRAPH PARTITIONING

Clustering on a large graph aims to partition the graph into several densely connected components. Identification of functional related protein modules in large protein–protein interaction networks, community detection in social networks, etc. are some of the applications of graph clustering. Many of the existing graph clustering methods mainly concentrate on the topological structure of a graph so that each partition achieves a cohesive internal structure.

7.4.1 Elements of Graph Theory

A *graph* $G = (V, E)$ consists of a vertex set V and an edge set E.

- If G is a *directed graph*, each edge is an ordered pair of vertices.
- A *bipartite graph* is one in which the vertices can be divided into two groups, so that all edges join vertices in different groups.

A hypergraph is a generalization of a graph wherein edges can connect more than two vertices and are called hyperedges. Graphs naturally represent many kinds of information in mathematical and computer science problems. Similarly, hypergraphs also arise naturally in important practical problems, including circuit layout and numerical linear algebra. Given a hypergraph H, *k-way partitioning* of H assigns vertices of H to k disjoint nonempty partitions. The k-way partitioning problem seeks to minimize a given cost function of such an assignment. *Net cut* is the standard cost function, which is the number of hyperedges that span more than one partition. Constraints are typically imposed on the solution and make the problem difficult.

7.5 HYPERGRAPH PARTITIONING

A hypergraph is a graph whose edges can connect more than two vertices called hyperedges. The clustering problem is then formulated as finding the minimum-cut of a hypergraph. A minimum-cut is the removal of the set of hyperedges (with minimum edge weight) that separates the hypergraph into k unconnected components. The minimum-cut of the hypergraph into k unconnected components gives the desired clustering. The HMETIS package is employed for partitioning. An advantage of this approach is that the clustering problem can be mapped to a graph problem without the explicit computation of similarity, which makes this approach computationally efficient with $O(n \times d \times k)$ assuming a (close to) linear performing hypergraph partition. Since in this formulation there is only a single weight associated with a hyperedge, sample-wise frequency information gets lost. Hypergraph-based clustering consists of the following steps:

1. Define the condition for connecting a number of objects (which will be the vertices of the hypergraph) by a hyperedge.
2. Define a measure of the strength or weight of a hyperedge.
3. Use a graph-partitioning algorithm to partition the hypergraph into two parts in such a way that the weight of the hyperedges cut is minimized.
4. Continue the partitioning until a fixed number of partitions are achieved, or until a "fitness" condition for the goodness of a cluster indicates that a current partition is a good cluster.

7.6 COBWEB CLUSTERING

COBWEB is an incremental hill-climbing strategy with bidirectional operators. It begins with a collection of unclassified objects and some means of measuring

the similarity of objects. COBWEB incrementally organizes observations into a classification tree. In the above algorithm, objects are treated by a distance function as a vector of n features.

It starts empty and creates a full concept hierarchy (classification tree) with each leaf representing a single instance/object. We can choose how deep we want to go in the tree hierarchy for the specific application at hand.

Objects are described as nominal attribute–value pairs. Each created node is a *probabilistic concept* (a class) that stores the probability of being matched (count/total), and for each attribute, the probability of being on, $P(a = v|C)$, only counts need be stored. Arcs in tree are just connections whereas the nodes store information across all attributes. Tradeoff between intraclass similarity and interclass dissimilarity: sums measures from each individual attribute.

Intraclass similarity is a function of $P(A_i = V_{ij}|C_k)$. Predictability of C is given as V; larger value of P means if class is C, A likely to be V. Objects within a class should have similar attributes.

Interclass dissimilarity is a function of $P(C_k|A_i = V_{ij})$; predictiveness of C is given as V. Larger value of P means $A = V$, which suggests instance is member of class C rather than some other class. A is a stronger predictor of class C.

7.7 FUTURE RESEARCH DIRECTIONS

Clustering algorithms can be used for detecting functional modules in protein interaction networks. Ant-based clustering algorithms have been used in a large variety of applications. They are applied for web usage mining. Clustering nowadays is proving to be good for the discrimination of non-rain weather and heavy rainstorm weather, but not effective for other weather types. It can be used for analyzing market basket, monitoring customer transactions, and producing specified results. Clustering and pattern discovery can also be used in very high-dimensional discrete-attribute data sets.

SUMMARY

Clustering is an unsupervised learning technique in which the output classes are unknown. It does not have a model. The input objects are just classified based on some observed criteria, which may not be predefined. It has got a wide variety of applications in bioinformatics, image classification, and so on. Weighted clustering on a large graph aims to partition the graph into several densely connected components. Typical applications of graph clustering include social networks, computational biology, and so on. DBSCAN is a density-based clustering that processes a set of data points; a point will either be in a cluster or will be classified as noise. These are different types of clustering, and they have got a wide variety of applications in data mining.