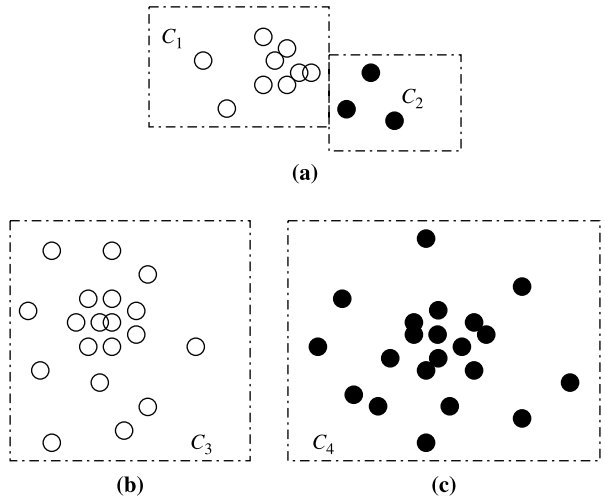

FIGURE 8.14

Merging clusters in probabilistic hierarchical clustering: (a) Merging clusters C_1 and C_2 leads to an increase in overall cluster quality, but merging clusters (b) C_3 and (c) C_4 does not.

Algorithm: A probabilistic hierarchical clustering algorithm.

Input:

- $D = \{\boldsymbol{o_1}, \ldots, \boldsymbol{o_n}\}$: a data set containing n objects;

Output: A hierarchy of clusters.
Method:

(1) **create** a cluster for each object $C_i = \{\boldsymbol{o_i}\}$, $1 \leq i \leq n$;
(2) **for** $i = 1$ to n
(3) **find** pair of clusters C_i and C_j such that $C_i, C_j = \arg\max_{i \neq j} \log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)}$;
(4) **if** $\log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)} > 0$ **then** merge C_i and C_j;
(5) **else** stop;

FIGURE 8.15

A probabilistic hierarchical clustering algorithm.

data set, there may exist multiple hierarchies that fit the observed data. Neither algorithmic approaches nor probabilistic approaches can find the distribution of such hierarchies. Recently, Bayesian tree-structured models have been developed to handle such problems. Bayesian and other sophisticated probabilistic clustering methods are considered advanced topics and are not covered in this book.

8.4 Density-based and grid-based methods

Most of the partitioning and hierarchical methods are designed to find spherical-shaped clusters. They have difficulty finding clusters of arbitrary shape such as the "S" shape and oval clusters in Fig. 8.16.

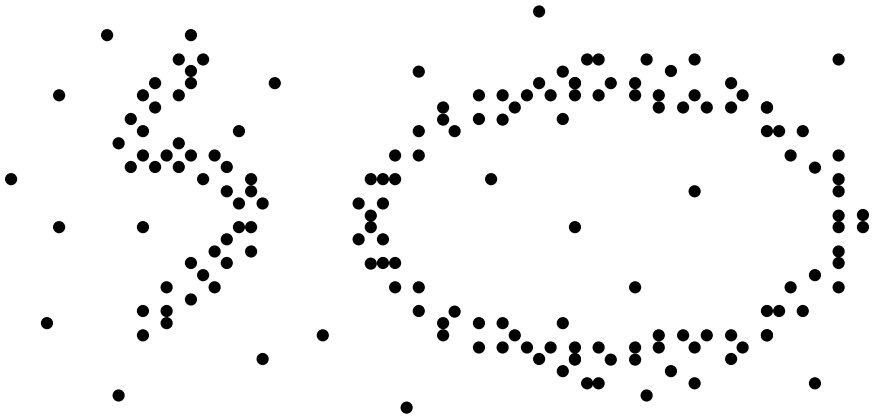

FIGURE 8.16

Clusters of arbitrary shape.

Although some feature transformation methods, such as kernel k-means, may help, it is often tricky to choose appropriate kernel functions. Given such data, they would likely inaccurately identify convex regions, where noises or outliers are included in the clusters.

To find clusters of arbitrary shape, alternatively, we can model clusters as dense regions in the data space, separated by sparse regions. This is the main strategy behind *density-based clustering methods*, which can discover clusters of nonspherical shape. In this section, you will learn the basic techniques of density-based clustering by studying two representative methods, namely, DBSCAN (Section 8.4.1) and DENCLUE (Section 8.4.2). To tackle the computational cost in density-based clustering, the data space may be partitioned into a grid. This idea motivates the grid-based clustering methods (Section 8.4.3).

8.4.1 DBSCAN: density-based clustering based on connected regions with high density

"How can we find dense regions in density-based clustering?" The *density* of an object $\boldsymbol{o}$ can be measured by the number of objects close to $\boldsymbol{o}$. **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) finds *core objects*, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters. In this section, let us explain **DBSCAN***, an improved version of the original **DBSCAN**.

"How does **DBSCAN*** *quantify the neighborhood of an object?"* **DBSCAN*** employs a user-specified parameter $\epsilon > 0$ to specify the radius of a neighborhood that is considered for every object. The **ϵ-neighborhood** of an object $\boldsymbol{o}$ is the space within a radius ϵ centered at $\boldsymbol{o}$.

Due to the fixed neighborhood size parameterized by ϵ, the **density of a neighborhood** can be measured simply by the number of objects in the neighborhood. To determine whether a neighborhood is dense or not, DBSCAN* uses another user-specified parameter, $MinPts$, which specifies the density

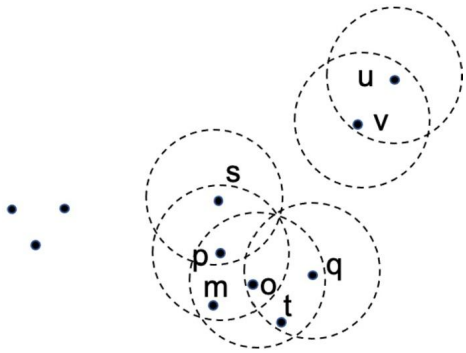

FIGURE 8.17

Density-reachability and density-connectivity in DBSCAN.

threshold of dense regions. An object is a **core object** if the ϵ-neighborhood of the object contains at least $MinPts$ objects, otherwise, it is **noise**. Core objects are the pillars of dense regions.

Given a set, D, of objects, we can identify all core objects with respect to the given parameters, ϵ and $MinPts$. The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters.

Two core objects $\boldsymbol{p}$ and $\boldsymbol{q}$ are **ϵ-reachable** if $d(\boldsymbol{p}, \boldsymbol{q}) \leq \epsilon$, that is, $\boldsymbol{p}$ is in the ϵ-neighbor of $\boldsymbol{q}$ and vice versa. Two core objects $\boldsymbol{p}$ and $\boldsymbol{q}$ are **density-connected** if $\boldsymbol{p}$ and $\boldsymbol{q}$ are ϵ-reachable or transitively ϵ-reachable, where $\boldsymbol{p}$ and $\boldsymbol{q}$ are transitive ϵ-reachable if there exist one or multiple core objects $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_l$ such that $\boldsymbol{p}$ and $\boldsymbol{r}_1$ are ϵ-reachable, $\boldsymbol{r}_i$ and $\boldsymbol{r}_{i+1}$ $(1 \leq i < l)$ are ϵ-reachable, and $\boldsymbol{r}_l$ and $\boldsymbol{q}$ are ϵ-reachable. Then, a cluster C with respect to parameters ϵ and $MinPts$ is simply a nonempty maximal subset of core objects that every pair of objects in C is density connected.

In DBSCAN*, a cluster contains only core objects. However, it is easy to assign a noncore object that is in the ϵ-neighborhood of a core object to the cluster containing that core object. DBSCAN explicitly identifies such noncore objects as **border objects**, and DBSCAN* can use a postprocessing step to pick up those border objects. Those objects that do not belong to any ϵ-neighborhood are outliers.

Example 8.7. Density-reachability and density-connectivity. Consider Fig. 8.17 for a given ϵ represented by the radius of the circles, and, say, let $MinPts = 3$.

Of the labeled objects, $\boldsymbol{p}$, $\boldsymbol{m}$, $\boldsymbol{o}$, $\boldsymbol{q}$, and $\boldsymbol{t}$ are core objects, since each of the ϵ-neighborhoods (dashed circles in the figure) of them contains at least three objects. Objects $\boldsymbol{p}$ and $\boldsymbol{o}$ are ϵ-reachable, so are $\boldsymbol{o}$ and $\boldsymbol{q}$. Thus $\boldsymbol{p}$ and $\boldsymbol{q}$ are density-connected.

It can be verified that the core objects $\boldsymbol{p}$, $\boldsymbol{m}$, $\boldsymbol{o}$, $\boldsymbol{q}$, and $\boldsymbol{t}$ form a cluster, since each two among them are density-connected and no other core objects can be added into this group so that the pairwise density-connectivity is maintained.

Object $\boldsymbol{s}$ is not a core object, since the ϵ-neighborhood of s contains only two objects. However, $\boldsymbol{s}$ is in the ϵ-neighborhood of core object $\boldsymbol{p}$, thus $\boldsymbol{s}$ is a border object.

Objects $\boldsymbol{u}$ and $\boldsymbol{v}$ are not core objects, and they do not belong to the ϵ-neighborhood of any core objects. Thus they are outliers. □

Algorithm: DBSCAN*: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- $MinPts$: the neighborhood density threshold.

Output: A set of density-based clusters.
Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ε-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ε-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the ε-neighborhood of p' has at least MinPts points,
                         add those points to N and add p' to C;
(12)         end for
(13)         output C;
(14)     else mark p as noise;
(15) until no object is unvisited;
```

FIGURE 8.18

DBSCAN* algorithm.

"How does DBSCAN find clusters?"* Initially, all objects in a given data set D are marked as "`unvisited`." DBSCAN* randomly selects an unvisited object $\boldsymbol{p}$, marks $\boldsymbol{p}$ as "`visited`," and checks whether the ϵ-neighborhood of $\boldsymbol{p}$ contains at least $MinPts$ objects. If not, $\boldsymbol{p}$ is marked as a `noise` point. Otherwise, a new cluster C is created for $\boldsymbol{p}$, and all the objects in the ϵ-neighborhood of $\boldsymbol{p}$ are added to a candidate set, N.

DBSCAN* iteratively adds to C those core objects in N that do not belong to any cluster. In this process, for an object $\boldsymbol{p'}$ in N that carries the label "`unvisited`," DBSCAN* marks it as "`visited`" and checks its ϵ-neighborhood. If the ϵ-neighborhood of $\boldsymbol{p'}$ has at least $MinPts$ objects, $\boldsymbol{p'}$ is labeled as a core object and added into C, those objects in the ϵ-neighborhood of $\boldsymbol{p'}$ are added to N. DBSCAN* continues adding objects to C until C can no longer be expanded, that is, N is empty. At this time, cluster C is completed, and thus is output.

To find the next cluster, DBSCAN* randomly selects an unvisited object from the remaining ones. The clustering process continues until all objects are visited. The pseudocode of the DBSCAN* algorithm is given in Fig. 8.18.

If a spatial index is used, the computational complexity of DBSCAN* is $O(n \log n)$, where n is the number of database objects. Otherwise, the complexity is $O(n^2)$. With appropriate settings of the user-defined parameters, ϵ and $MinPts$, the algorithm is effective in finding arbitrary-shaped clusters.

It is not easy to specify two parameters, ϵ and $MinPts$, in DBSCAN. Moreover, density-based clusters may also have hierarchies. For example, within a dense area there may be a sub-area is sub-*

stantially denser. Can we find hierarchical density-based clusters? Indeed, DBSCAN* can be extended to HDBSCAN*, which achieves density-based hierarchical clustering.

HDBSCAN* only takes one parameter, $MinPts$. For an object $\boldsymbol{p}$, the **core distance** of $\boldsymbol{p}$, denoted by $d_{core}(\boldsymbol{p})$ is the distance from $\boldsymbol{p}$ to its $MinPts$th nearest neighbor (including $\boldsymbol{p}$ itself). In other words, $d_{core}(\boldsymbol{p})$ is the minimum radius with respect to which $\boldsymbol{p}$ is a core object in DBSCAN*. For two objects $\boldsymbol{p}$ and $\boldsymbol{q}$, the **mutual reachability distance** between them is $d_{mreach}(\boldsymbol{p}, \boldsymbol{q}) = \max\{d_{core}(\boldsymbol{p}), d_{core}(\boldsymbol{q}), d(\boldsymbol{p}, \boldsymbol{q})\}$. In other words, $d_{mreach}(\boldsymbol{p}, \boldsymbol{q})$ is the minimum radius ϵ such that $\boldsymbol{p}$ and $\boldsymbol{q}$ are ϵ-reachable in DBSCAN*.

Given a set of objects as the input to HDBSCAN*, we can construct a **mutual reachability graph** G_{MinPts}, which is a complete graph. Every object in the input is a node in the mutual reachability graph. The weight of the edge between $\boldsymbol{p}$ and $\boldsymbol{q}$ is the mutual reachability distance $d_{mreach}(\boldsymbol{p}, \boldsymbol{q})$. We can apply the minimum spanning tree approach (Section 8.3.3) on the mutual reachability graph to find density-based hierarchical clusters.

To further reduce the demand of setting parameters, a cluster analysis method called **OPTICS** was proposed. OPTICS does not explicitly produce a data set clustering. Instead, it outputs a **cluster ordering**, a linear list of all objects under analysis, representing the *density-based clustering structure* of the data. Objects in a denser cluster are listed closer to each other in the cluster ordering. This ordering is equivalent to density-based clustering obtained from a wide range of parameter settings. Thus, OPTICS does not require the user to provide a specific density threshold. The cluster ordering can be used to extract basic clustering information (e.g., cluster centers, or arbitrary-shaped clusters), derive the intrinsic clustering structure, and provide a visualization of the clustering.

To construct the different clusterings simultaneously, the objects are processed in a specific order. This order selects an object that is density-reachable with respect to the lowest ϵ value so that clusters with higher density (i.e., lower ϵ) will be finished first. For example, Fig. 8.19 shows the reachability plot for a simple 2-D data set, which presents a general overview of how the data are structured and clustered. The data objects are plotted in the clustering order (horizontal axis) together with their respective reachability-distances (vertical axis). The three Gaussian "bumps" in the plot reflect three clusters in the data set.

OPTICS can be seen as a generalization of DBSCAN that replaces the ϵ parameter with a maximum value that mostly affects performance. $MinPts$ then essentially becomes the minimum cluster size to find. While the algorithm is much easier to parameterize than DBSCAN, it usually produces a hierarchical clustering instead of the simple data partitioning that DBSCAN produces.

8.4.2 DENCLUE: clustering based on density distribution functions

Density estimation is a core issue in density-based clustering. **DENCLUE** (DENsity-based CLUsteRing) is a clustering method based on a set of density distribution functions. We first give some background on density estimation and then describe the DENCLUE algorithm.

In probability and statistics, **density estimation** is the estimation of an unobservable underlying probability density function based on a set of observed data. In the context of density-based clustering, the unobservable underlying probability density function is the true distribution of the population of all possible objects to be analyzed. The observed data set is regarded as a random sample from that population.

In DENCLUE, **kernel density estimation** is used, which is a nonparametric density estimation approach from statistics. The general idea behind kernel density estimation is simple. We treat an observed

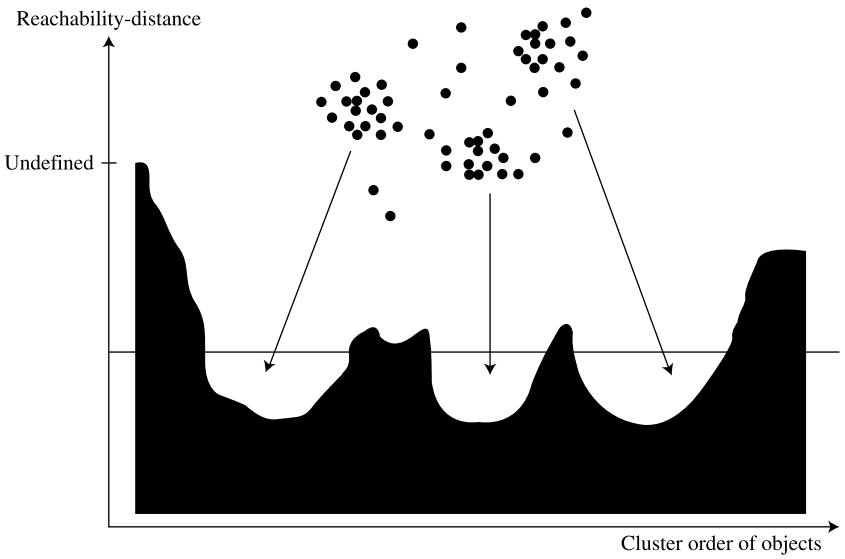

FIGURE 8.19

Cluster ordering in OPTICS. *Source:* Adapted from Ankerst, Breunig, Kriegel, and Sander [ABKS99].

object as an indicator of high-probability density in the surrounding region. The probability density at a point depends on the distances from this point to the observed objects.

Formally, let $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ be an independent and identically distributed sample of a random variable f. The *kernel density approximation of the probability density function* is

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x_i}}{h}\right), \tag{8.22}$$

where $K()$ is a kernel function and h is the bandwidth serving as a smoothing parameter. A **kernel** can be regarded as a function modeling the influence of a sample point within its neighborhood. Technically, a kernel $K()$ is a nonnegative real-valued integrable function that should satisfy two requirements: $\int_{-\infty}^{+\infty} K(u)du = 1$ and $K(-u) = K(u)$ for all values of u. A frequently used kernel is a standard Gaussian function with a mean of 0 and a variance of 1:

$$K\left(\frac{\boldsymbol{x} - \boldsymbol{x_i}}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x - x_i)^2}{2h^2}}. \tag{8.23}$$

DENCLUE uses a Gaussian kernel to estimate density based on the given set of objects to be clustered. A point $\boldsymbol{x^*}$ is called a **density attractor** if it is a local maximum of the estimated density function. To avoid trivial local maximum points, DENCLUE uses a noise threshold, ξ, and only considers those density attractors $\boldsymbol{x^*}$ such that $\hat{f}(\boldsymbol{x^*}) \geq \xi$. These nontrivial density attractors are the centers of clusters.

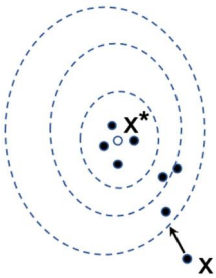

FIGURE 8.20

Hill-climbing in DENCLUE.

The objects under analysis are assigned to clusters through density attractors using a stepwise hill-climbing procedure. For an object, $\boldsymbol{x}$, the hill-climbing procedure starts from $\boldsymbol{x}$ and is guided by the gradient of the estimated density function. That is, the density attractor for $\boldsymbol{x}$ is computed as

$$\begin{aligned} \boldsymbol{x^0} &= \boldsymbol{x} \\ \boldsymbol{x^{j+1}} &= \boldsymbol{x^j} + \delta \frac{\nabla \hat{f}(\boldsymbol{x^j})}{|\nabla \hat{f}(\boldsymbol{x^j})|}, \end{aligned} \tag{8.24}$$

where δ is a parameter to control the speed of convergence, and

$$\nabla \hat{f}(\boldsymbol{x}) = \frac{1}{h^{d+2} n \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x}-\boldsymbol{x_i}}{h}\right)(\boldsymbol{x_i}-\boldsymbol{x})}. \tag{8.25}$$

The hill-climbing procedure stops at step $k > 0$ if $\hat{f}(\boldsymbol{x^{k+1}}) < \hat{f}(\boldsymbol{x^k})$, and assigns $\boldsymbol{x}$ to the density attractor $\boldsymbol{x^*} = \boldsymbol{x^k}$. An object $\boldsymbol{x}$ is an outlier or noise if it converges in the hill-climbing procedure to a local maximum $\boldsymbol{x^*}$ with $\hat{f}(\boldsymbol{x^*}) < \xi$.

Fig. 8.20 illustrates the hill-climbing idea. For a point $\boldsymbol{x}$, the density attractor for x is initialized to $\boldsymbol{x^0} = \boldsymbol{x}$. In the next iteration, the density attractor moves a small step towards the direction indicated by the gradient of the density function, shown by the arrow in the figure, until the point $\boldsymbol{x}^*$ where density is stable (the white circle point in the figure), which is a local optimal.

A cluster in DENCLUE is a set of density attractors X and a set of input objects C such that each object in C is assigned to a density attractor in X, and there exists a path between every pair of density attractors where the density is above ξ. By using multiple density attractors connected by paths, DENCLUE can find clusters of arbitrary shape.

DENCLUE has several advantages. It can be regarded as a generalization of several well-known clustering methods such as single-linkage approaches and DBSCAN. Moreover, DENCLUE is invariant against noise. The kernel density estimation can effectively reduce the influence of noise by uniformly distributing noise into the input data.

8.4.3 Grid-based methods

As analyzed in the previous subsections, computing density for density-based clustering may be costly, particularly on large data sets and data sets of high dimensionality. To tackle the efficiency and scalability challenges, one idea is to partition the data space into cells using a grid. This motivates the grid-based clustering methods.

The *grid-based clustering* approach uses a multiresolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed. The main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space.

Typically, grid-based clustering takes three steps.

1. We create a grid structure so that the data space is partitioned into a finite number of cells.
2. For each cell, we calculate the cell density. By a carefully designed method, we may be able to scan the data once and derive the densities for all cells. This step is a key to gain efficiency and scalability.
3. We use the dense cells to assemble clusters and optionally summarize dense cells and the corresponding clusters.

Let us illustrate this using an example. **CLIQUE** (CLustering In QUEst) is a simple grid-based method for finding density-based clusters in subspaces. CLIQUE partitions each dimension into nonoverlapping intervals, thereby partitioning the entire data space into cells. It uses a density threshold to identify *dense* cells and *sparse* ones. A cell is dense if the number of objects mapped to it exceeds the density threshold.

The main strategy behind CLIQUE for identifying a candidate search space uses the monotonicity of dense cells with respect to dimensionality. This is based on the *Apriori property* used in frequent pattern and association rule mining (Chapter 4). In the context of clusters in subspaces, the monotonicity says the following. A k-dimensional cell c ($k > 1$) can have at least l points only if every $(k-1)$-dimensional projection of c, which is a cell in a $(k-1)$-dimensional subspace, has at least l points. Consider Fig. 8.21, where the data space contains three dimensions: *age, salary*, and *vacation*. A 2-D

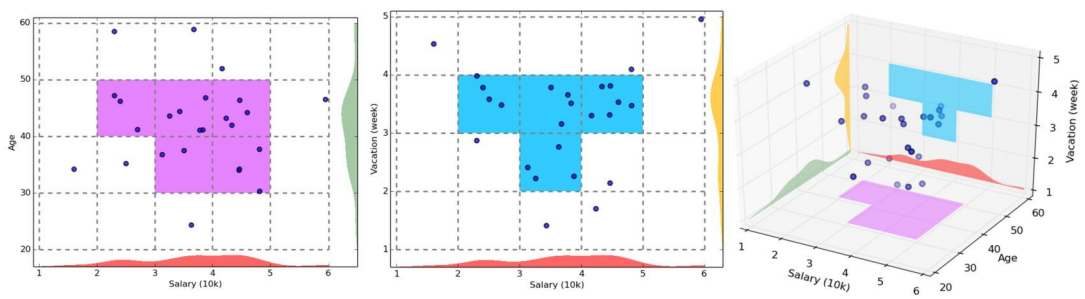

FIGURE 8.21

Dense units found with respect to *age* for the dimensions *salary* and *vacation* are intersected to provide a candidate search space for dense units of higher dimensionality.

cell, say in the subspace formed by *age* and *salary*, contains l points only if the projection of this cell in every dimension, that is, *age* and *salary*, respectively, contains at least l points.

CLIQUE performs clustering in three steps. In the first step, CLIQUE partitions the d-dimensional data space into nonoverlapping rectangular units.

In the second step, CLIQUE identifies the dense units among these. CLIQUE finds dense cells in all of the subspaces. To do so, CLIQUE partitions every dimension into intervals, and identifies intervals containing at least l points, where l is the density threshold. CLIQUE then iteratively joins two k-dimensional dense cells, c_1 and c_2, in subspaces $(D_{i_1}, \ldots, D_{i_k})$ and $(D_{j_1}, \ldots, D_{j_k})$, respectively, if $D_{i_1} = D_{j_1}, \ldots, D_{i_{k-1}} = D_{j_{k-1}}$, and c_1 and c_2 share the same intervals in those dimensions. The join operation generates a new $(k+1)$-dimensional candidate cell c in space $(D_{i_1}, \ldots, D_{i_{k-1}}, D_{i_k}, D_{j_k})$. CLIQUE checks whether the number of points in c passes the density threshold. The iteration terminates when no candidates can be generated or no candidate cells are dense.

In the last step, CLIQUE uses the dense cells in each subspace to assemble clusters, which can be of arbitrary shape. The idea is to apply the Minimum Description Length (MDL) principle (Chapter 7) to use the *maximal regions* to cover connected dense cells, where a maximal region is a hyperrectangle where every cell falling into this region is dense, and the region cannot be extended further in any dimension in the subspace. Finding the best description of a cluster in general is NP-hard. Thus CLIQUE adopts a simple greedy approach. It starts with an arbitrary dense cell, finds a maximal region covering the cell, and then works on the remaining dense cells that have not yet been covered. The greedy method terminates when all dense cells are covered.

"How effective is CLIQUE?" CLIQUE automatically finds subspaces of the highest dimensionality such that high-density clusters exist in those subspaces. It is insensitive to the order of input objects and does not presume any canonical data distribution. It scales linearly with the size of the input and has good scalability as the number of dimensions in the data increases. However, obtaining a meaningful clustering is dependent on proper tuning of the grid size (which is a stable structure here) and the density threshold. This can be difficult in practice because the grid size and density threshold are used across all combinations of dimensions in the data set. Thus the accuracy of the clustering results may be degraded at the expense of the method's simplicity. Moreover, for a given dense region, all projections of the region onto lower-dimensionality subspaces will also be dense. This can result in a large overlap among the reported dense regions. Furthermore, it is difficult to find clusters of rather different densities within different dimensional subspaces.

STING is another representative grid-based multiresolution clustering technique. In STING, the spatial area of the input objects is divided into rectangular cells. The space can be divided in a hierarchical and recursive way. Several levels of such rectangular cells correspond to different levels of resolution and form a hierarchical structure: Each cell at a high level is partitioned to form a number of cells at the next lower level. Statistical information regarding the attributes in each grid cell, such as the mean, maximum, and minimum values, is precomputed and stored as *statistical parameters*. These statistical parameters are useful for query processing and for other data analysis tasks.

Fig. 8.22 shows a hierarchical structure for STING clustering. The statistical parameters of higher-level cells can easily be computed from the parameters of the lower-level cells. These parameters include the following: the attribute-independent parameter, *count*; and the attribute-dependent parameters, *mean*, $stdev$ (standard deviation), min (minimum), max (maximum), and the type of *distribution* that the attribute value in the cell follows such as *normal, uniform*, *exponential*, or *none* (if the distribution is unknown). Here, the attribute is a selected measure for analysis such as *price* for house objects.

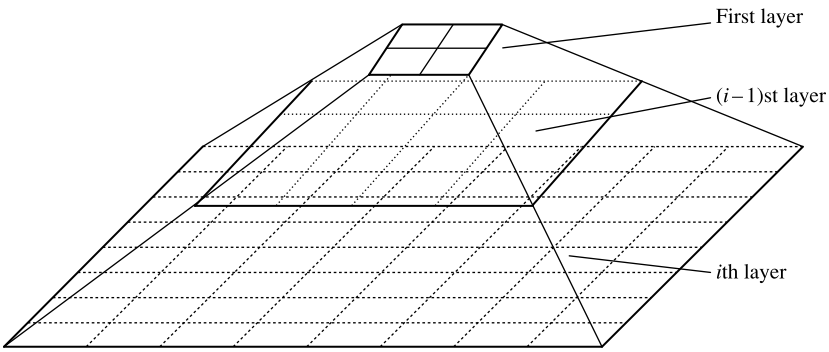

FIGURE 8.22

Hierarchical structure for STING clustering.

When the data are loaded into the database, the parameters *count*, $mean$, $stdev$, min, and max of the bottom-level cells are calculated directly from the data. The value of *distribution* may either be assigned by the user if the distribution type is known beforehand or obtained by hypothesis tests such as the χ^2 test. The type of distribution of a higher-level cell can be computed based on the majority of distribution types of its corresponding lower-level cells in conjunction with a threshold filtering process. If the distributions of the lower-level cells disagree with each other and fail the threshold test, the distribution type of the high-level cell is set to *none*.

"How is this statistical information useful for query answering?" The statistical parameters can be used in a top-down, grid-based manner as follows. First, a layer within the hierarchical structure is determined from which the query-answering process is to start. This layer typically contains a small number of cells. For each cell in the current layer, we compute the confidence interval (or estimated probability range) reflecting the relevancy of the cell to the given query. The irrelevant cells are removed from further consideration. Processing of the next lower level examines only the remaining relevant cells. This process repeats until the bottom layer is reached. At this time, if the query specification is met, the regions of relevant cells that satisfy the query are returned. Otherwise, the data points that fall into the relevant cells are retrieved and further processed until they meet the query's requirements.

An interesting property of STING is that it approaches the clustering result of DBSCAN if the granularity approaches 0 (i.e., toward very low-level data). In other words, using the count and cell size information, dense clusters can be identified approximately using STING. Therefore STING can also be regarded as a density-based clustering method.

"What advantages does STING offer over other clustering methods?" STING offers several advantages. First, the grid-based computation is *query-independent* because the statistical information stored in each cell represents the summary information of the data in the grid cell, independent of the query. Moreover, the grid structure facilitates parallel processing and incremental updating. Last, the efficiency of STING is a major advantage: STING goes through the database once to compute the statistical parameters of the cells and hence the time complexity of generating clusters is $O(n)$, where n is the total number of objects. After generating the hierarchical structure, the query processing time

is $O(g)$, where g is the total number of grid cells at the lowest level, which is usually much smaller than n.

Because STING uses a multiresolution approach to cluster analysis, the quality of STING clustering depends on the granularity of the lowest level of the grid structure. If the granularity is very fine, the cost of processing increases substantially; however, if the bottom level of the grid structure is too coarse, it may reduce the quality of cluster analysis. Moreover, STING does not consider the spatial relationship between the children and their neighboring cells for construction of a parent cell. As a result, the shapes of the resulting clusters are isothetic, that is, all the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected. This may lower the quality and accuracy of the clusters despite the fast processing time of the technique.

8.5 Evaluation of clustering

By now you have learned what clustering is and know several popular clustering methods. You may ask, "*When I try out a clustering method on a data set, how can I evaluate whether the clustering results are good?*" In general, *cluster evaluation* assesses the feasibility of clustering analysis on a data set and the quality of the results generated by a clustering method. The major tasks of clustering evaluation include the following:

- *Assessing clustering tendency*. In this task, for a given data set, we assess whether a nonrandom structure exists in the data. Blindly applying a clustering method on a data set will return clusters; however, the clusters mined may be misleading. Clustering analysis on a data set is meaningful only when there is a nonrandom structure in the data.
- *Determining the number of clusters in a data set*. A few algorithms, such as k-means, require the number of clusters in a data set as the parameter. Moreover, the number of clusters can be regarded as an interesting and important summary statistic of a data set. Therefore it is desirable to estimate this number even before a clustering algorithm is used to derive detailed clusters.
- *Measuring clustering quality*. After applying a clustering method on a data set, we want to assess how good the resulting clusters are. A number of measures can be used. Some methods measure how well the clusters fit the data set, while others measure how well the clusters match the ground truth, if such truth is available. There are also measures that score clusterings and thus can compare two sets of clustering results on the same data set.

In this section, we discuss these three topics one by one.

8.5.1 Assessing clustering tendency

Clustering tendency assessment determines whether a given data set has a nonrandom structure, which may lead to meaningful clusters. Consider a data set that does not have any nonrandom structure, such as a set of uniformly distributed points in a data space. Even though a clustering algorithm may return clusters for the data, those clusters are random and thus are not meaningful.

Example 8.8. Clustering requires nonuniform distribution of data. Fig. 8.23 shows a data set that is uniformly distributed in 2-D data space. Although a clustering algorithm may still artificially partition

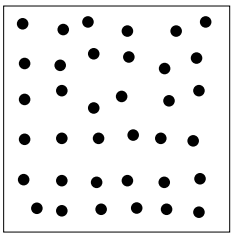

FIGURE 8.23

A data set that is uniformly distributed in the data space.

the points into groups, the groups will unlikely mean anything significant to the application due to the uniform distribution of the data. □

"How can we assess the clustering tendency of a data set?" Intuitively, we can try to measure the probability that the data set is generated by a uniform data distribution. This can be achieved using statistical tests for spatial randomness. To illustrate this idea, let us look at a simple yet effective statistic called the Hopkins statistic.

The **Hopkins Statistic** is a spatial statistic that tests the spatial randomness of a variable as distributed in a space. Given a data set, D, which is regarded as a sample of a random variable, o, we want to determine how far away o is from being uniformly distributed in the data space. We calculate the Hopkins Statistic as follows:

1. Sample n points, $\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}$ from the data space. For each point, $\boldsymbol{p_i}$ $(1 \leq i \leq n)$, we find the nearest neighbor in D, and let x_i be the distance between $\boldsymbol{p_i}$ and its nearest neighbor in D. That is,

$$x_i = \min_{\boldsymbol{v} \in D} \{dist(\boldsymbol{p_i}, \boldsymbol{v})\}. \tag{8.26}$$

2. Sample n points, $\boldsymbol{q_1}, \ldots, \boldsymbol{q_n}$ uniformly from D without replacement. That is, each point in D has the same probability of being included in this sample, and one point can only be included in the sample at most once. For each $\boldsymbol{q_i}$ $(1 \leq i \leq n)$, we find the nearest neighbor of $\boldsymbol{q_i}$ in $D - \{\boldsymbol{q_i}\}$, and let y_i be the distance between $\boldsymbol{q_i}$ and its nearest neighbor in $D - \{\boldsymbol{q_i}\}$. That is,

$$y_i = \min_{\boldsymbol{v} \in D, \boldsymbol{v} \neq \boldsymbol{q_i}} \{dist(\boldsymbol{q_i}, \boldsymbol{v})\}. \tag{8.27}$$

3. Calculate the Hopkins statistic, H, as

$$H = \frac{\sum_{i=1}^{n} x_i^d}{\sum_{i=1}^{n} x_i^d + \sum_{i=1}^{n} y_i^d}, \tag{8.28}$$

where d is the dimensionality of the data set D.

"What does the Hopkins statistic tell us about how likely data set D *follows a uniform distribution in the data space?"* If D is uniformly distributed, then $\sum_{i=1}^{n} y_i^d$ and $\sum_{i=1}^{n} x_i^d$ are close to each other,

and thus H tends to be about 0.5. However, if D is highly skewed, then the points in D are closer to their nearest neighbors than the random points $\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}$ are, and thus $\sum_{i=1}^{n} x_i^d$ shall be substantially larger than $\sum_{i=1}^{n} y_i^d$ in expectation, and H tends to be close to 1.

Example 8.9. Hopkins statistic. Consider a 1-D data set $D = \{0.9, 1, 1.3, 1.4, 1.5, 1.8, 2, 2.1, 4.1, 7, 7.4, 7.5, 7.7, 7.8, 7.9, 8.1\}$ in the data space $[0, 10]$. We draw a sample of four points from D without replacement, say, 1.3, 1.8, 7.5, and 7.9. We also draw a sample of four points uniformly from the data space $[0, 10]$, say, 1.9, 4, 6, 8. Then, the Hopkins statistic can be calculated as

$$H = \frac{|1.9-2| + |4-4.1| + |6-7| + |8-8.1|}{(|1.9-2| + |4-4.1| + |6-7| + |8-8.1|) + (|1.3-1.4| + |1.8-2| + |7.5-7.4| + |7.9-7.8|)}$$
$$= \frac{1.3}{1.3+0.5} = \frac{1.3}{1.8} = 0.72.$$

Since the Hopkins statistic is substantially larger than 0.5 and is close to 1, the data set D has a strong clustering tendency. Indeed, there are two clusters, one around 1.5 and the other one around 7.8. □

In addition to Hopkins statistic, there are some other methods, such as spatial histogram and distance distribution, comparing statistics between a data set under clustering tendency analysis and the corresponding uniform distribution. For example, distance distribution compares the distribution of pairwise distance in the target data set and that in a random uniform sample from the data space.

8.5.2 Determining the number of clusters

Determining the "right" number of clusters in a data set is important, not only because some clustering algorithms like k-means require such a parameter, but also because the appropriate number of clusters controls the proper granularity of cluster analysis. It can be regarded as finding a good balance between *compressibility* and *accuracy* in cluster analysis. Consider two extreme cases. What if you were to treat the entire data set as a cluster? This would maximize the compression of the data, but such a cluster analysis has no value. In contrast, treating each object in a data set as a cluster gives the finest clustering resolution (i.e., most accurate due to the zero distance between an object and the corresponding cluster center). In some methods like k-means, this even achieves the minimum cost. However, having one object per cluster does not enable any data summarization.

Determining the number of clusters is far from easy, often because the "right" number is ambiguous. Figuring out the right number of clusters often depends on the distribution's shape and scale in the data set, as well as the clustering resolution required by the user. There are many possible ways to estimate the number of clusters.

For example, a simple method is to set the number of clusters to about $\sqrt{\frac{n}{2}}$ for a data set of n points. In expectation, each cluster has $\sqrt{2n}$ points. Section 8.2.2 introduces the Calinski-Harabasz index, which estimates the number of clusters for k-means.

Let us look at two more alternative methods.

The **elbow method** is based on the observation that increasing the number of clusters can help to reduce the sum of within-cluster variance of each cluster. This is because having more clusters allows one to capture finer groups of data objects that are more similar to each other. However, the marginal

effect of reducing the sum of within-cluster variances may drop if too many clusters are formed, because splitting a cohesive cluster into two gives only a small reduction. Consequently, a heuristic for selecting the right number of clusters is to use the turning point in the curve of the sum of within-cluster variances with respect to the number of clusters.

Technically, given a number, $k > 0$, we can form k clusters on the data set in question using a clustering algorithm like k-means, and calculate the sum of within-cluster variances, $var(k)$. We can then plot the curve of var with respect to k. The first (or most significant) turning point of the curve suggests the "right" number.

More advanced methods can determine the number of clusters using information criteria or information theoretic approaches. Please refer to the bibliographic notes for further information (Section 8.8).

The "right" number of clusters in a data set can also be determined by **cross-validation**, a technique often used in classification (Chapter 6). First, we divide the given data set, D, into m parts. Next, we use $m - 1$ parts to build a clustering model, and use the remaining part to test the quality of the clustering. For example, for each point in the test set, we can find the closest centroid. Consequently, we can use the sum of the squared distances between all points in the test set and the closest centroids to measure how well the clustering model fits the test set. For any integer $k > 0$, we repeat this process m times to derive clusterings of k clusters, using each part in turn as the test set. The average of the quality measure is taken as the overall quality measure. We can then compare the overall quality measure with respect to different values of k and find the number of clusters that best fits the data.

8.5.3 Measuring clustering quality: extrinsic methods

Suppose you have assessed the clustering tendency of a given data set. You may have also tried to predetermine the number of clusters in the set. You can now apply one or multiple clustering methods to obtain clusterings of the data set. "*How good is the clustering generated by a method, and how can we compare the clusterings generated by different methods?*"

Extrinsic vs. intrinsic methods

We have a few methods to choose from for measuring the quality of a clustering. In general, these methods can be categorized into two groups according to whether ground truth is available. Here, *ground truth* is the ideal clustering that is often built using human experts.

If ground truth is available, it can be used by the **extrinsic methods**, which compare the clustering against the ground truth and measure. If the ground truth is unavailable, we can use the **intrinsic methods**, which evaluate the goodness of a clustering by considering how well the clusters are separated. Ground truth can be considered as supervision in the form of "cluster labels." Hence, extrinsic methods are also known as *supervised methods*, whereas intrinsic methods are *unsupervised methods*.

In this section, we focus on extrinsic methods. We will discuss intrinsic methods in the next section.

Desiderata of extrinsic methods

When the ground truth is available, we can compare it with a clustering to assess the quality of the clustering. Thus the core task in extrinsic methods is to assign a score, $Q(\mathcal{C}, \mathcal{C}_g)$, to a clustering, $\mathcal{C}$, given the ground truth, $\mathcal{C}_g$. Whether an extrinsic method is effective largely depends on the measure, Q, it uses.

In general, a measure Q on clustering quality is effective if it satisfies the following four essential criteria:

- **Cluster homogeneity.** This requires that the purer the clusters in a clustering are, the better the clustering. Suppose that the ground truth says that the objects in a data set, $D = \{a, b, c, d, e, f, g, h\}$, can belong to three categories. Objects a and b are in category L_1, objects c and d belong to category L_2, and the others are in category L_3. Consider clustering, $\mathcal{C}_1 = \{\{a, b, c, d\}, \{e, f, g, h\}\}$, wherein a cluster $\{a, b, c, d\} \in \mathcal{C}_1$ contains objects from two categories, L_1 and L_2. Also consider clustering $\mathcal{C}_2 = \{\{a, b\}, \{c, d\}, \{e, f, g, h\}\}$, which is identical to $\mathcal{C}_1$ except that $\mathcal{C}_2$ is split into two clusters containing the objects in L_1 and L_2, respectively. A clustering quality measure, Q, respecting cluster homogeneity should give a higher score to $\mathcal{C}_2$ than $\mathcal{C}_1$, that is, $Q(\mathcal{C}_2, \mathcal{C}_g) > Q(\mathcal{C}_1, \mathcal{C}_g)$.
- **Cluster completeness.** This is the counterpart of cluster homogeneity. Cluster completeness requires that for a clustering, if any two objects belong to the same category according to the ground truth, then they should be assigned to the same cluster. Cluster completeness requires that a clustering should assign objects belonging to the same category (according to the ground truth) to the same cluster. Continue our previous example. Suppose clustering $\mathcal{C}_3 = \{\{a, b\}, \{c, d\}, \{e, f\}, \{g, h\}\}$. $\mathcal{C}_3$ and $\mathcal{C}_2$ are identical except that $\mathcal{C}_3$ split the objects in category L_3 into two clusters. Then, a clustering quality measure, Q, respecting cluster completeness should give a higher score to $\mathcal{C}_2$, that is, $Q(\mathcal{C}_2, \mathcal{C}_g) > Q(\mathcal{C}_1, \mathcal{C}_g)$.
- **Rag bag.** In many practical scenarios, there is often a "rag bag" category containing objects that cannot be merged with other objects. Such a category is often called "miscellaneous," "other," and so on. The rag bag criterion states that putting a heterogeneous object into a pure cluster should be penalized more than putting it into a rag bag. Consider a clustering $\mathcal{C}_1$ and a cluster $C \in \mathcal{C}_1$ such that all objects in C except for one, denoted by $\boldsymbol{o}$, belong to the same category according to the ground truth. Consider a clustering $\mathcal{C}_2$ identical to $\mathcal{C}_1$ except that $\boldsymbol{o}$ is assigned to a cluster $C' \neq C$ in $\mathcal{C}_2$ such that C' contains objects from various categories according to ground truth, and thus is noisy. In other words, C' in $\mathcal{C}_2$ is a rag bag. Then, a clustering quality measure Q respecting the rag bag criterion should give a higher score to $\mathcal{C}_2$, that is, $Q(\mathcal{C}_2, \mathcal{C}_g) > Q(\mathcal{C}_1, \mathcal{C}_g)$.
- **Small cluster preservation.** If a small category is split into small pieces in a clustering, those small pieces may likely become noise and thus the small category cannot be discovered from the clustering. The small cluster preservation criterion states that splitting a small category into pieces is more harmful than splitting a large category into pieces. Consider an extreme case. Let D be a data set of $n + 2$ objects such that, according to ground truth, n objects, denoted by $\boldsymbol{o_1}, \ldots, \boldsymbol{o_n}$, belong to one category and the other two objects, denoted by $\boldsymbol{o_{n+1}}, \boldsymbol{o_{n+2}}$, belong to another category. Suppose clustering $\mathcal{C}_1$ has three clusters, $C_1^1 = \{\boldsymbol{o_1}, \ldots, \boldsymbol{o_n}\}$, $C_2^1 = \{\boldsymbol{o_{n+1}}\}$, and $C_3^1 = \{\boldsymbol{o_{n+2}}\}$. Let clustering $\mathcal{C}_2$ have three clusters, too, namely $C_1^2 = \{\boldsymbol{o_1}, \ldots, \boldsymbol{o_{n-1}}\}$, $C_2^2 = \{\boldsymbol{o_n}\}$, and $C_3^2 = \{\boldsymbol{o_{n+1}}, \boldsymbol{o_{n+2}}\}$. In other words, $\mathcal{C}_1$ splits the small category and $\mathcal{C}_2$ splits the big category. A clustering quality measure Q preserving small clusters should give a higher score to $\mathcal{C}_2$, that is, $Q(\mathcal{C}_2, \mathcal{C}_g) > Q(\mathcal{C}_1, \mathcal{C}_g)$.

Categories of extrinsic methods

The ground truth may be used in different ways to evaluate clustering quality, which lead to different extrinsic methods. In general, the extrinsic methods can be categorized according to how the ground truth is used as follows.

- **The matching-based methods** examine how well the clustering results match the ground truth in partitioning the objects in the data set. For example, the purity methods assess how a cluster matches only those objects in one group in the ground truth.
- **The information theory–based methods** compare the distribution of the clustering results and that of the ground truth. Entropy or other measures in information theory are often employed to quantify the comparison. For example, we can measure the conditional entropy between the clustering results and the ground truth to measure whether there exists dependency between the information of the clustering results and the ground truth. The higher the dependency, the better the clustering results.
- **The pairwise comparison–based methods** treat each group in the ground truth as a class and then check the pairwise consistency of the objects in the clustering results. The clustering results are good if more pairs of objects of the same class are put into the same cluster, less pairs of objects of different classes are put into the same cluster, and less pairs of objects of the same class are put into different clusters.

Next, let us use some examples to illustrate the above categories of extrinsic methods.

Matching-based methods

The matching-based methods compare clusters in the clustering results and the groups in the ground truth. Let us use an example to explain the ideas.

Suppose a clustering method partitions a set of objects $D = \{o_1, \ldots, o_n\}$ into clusters $\mathcal{C} = \{C_1, \ldots, C_m\}$. The ground truth $\mathcal{G}$ also partitions the same set of objects into groups $\mathcal{G} = \{G_1, \ldots, G_l\}$. Let $C(o_x)$ and $G(o_x)$ $(1 \leq x \leq n)$ be the cluster-id and the group-id of object o_x in the clustering results and the ground truth, respectively.

For a cluster C_i $(1 \leq i \leq m)$, how well C_i matches group G_j in the ground truth can be measured by $|C_i \cap G_j|$, the larger the better. $\frac{|C_i \cap G_j|}{|C_i|}$ can be regarded as the purity of cluster C_i, where G_j matching C_i maximizes $|C_i \cap G_j|$. The purity of the whole clustering results can be calculated as the weighted sum of the purity of the clusters. That is,

$$purity = \sum_{i=1}^{m} \frac{|C_i|}{n} \max_{j=1}^{l} \{\frac{|C_i \cap G_j|}{|C_i|}\} = \frac{1}{n} \sum_{i=1}^{m} \max_{j=1}^{l} \{|C_i \cap G_j|\}. \tag{8.29}$$

The higher the purity, the purer are the clusters, that is, the more objects in each cluster belong to the same group in the ground truth. When the purity is 1, each cluster either matches a group perfectly or is a subset of a group. In other words, no two objects belong to two groups are mixed in one cluster. However, it is possible that multiple clusters partition a group in the ground truth.

Example 8.10. Purity. Consider the set of objects $D = \{a, b, c, d, e, f, g, h, i, j, k\}$. The clustering ground truth and two clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$ output by two methods are shown in Table 8.1.

The purity of clustering $\mathcal{C}_1$ is calculated by $\frac{1}{11} \times (4 + 2 + 4 + 1) = \frac{11}{11} = 1$ and that of clustering $\mathcal{C}_2$ is $\frac{1}{11}(2 + 3 + 1) = \frac{6}{11}$. In terms of purity, $\mathcal{C}_1$ is better than $\mathcal{C}_2$. Please note that, although $\mathcal{C}_1$ has purity 1, it splits G_1 in the ground truth into two clusters, C_1 and C_2. □

There are some other matching based methods further refine the measurement of matching quality, such as maximum matching and using F-measure.

Table 8.1 A set of objects, the clustering ground truth, and two clusterings.

Object	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*
Ground truth $\mathcal{G}$	G_1	G_1	G_1	G_1	G_1	G_1	G_2	G_2	G_2	G_2	G_3
Clustering $\mathcal{C}_1$	C_1	C_1	C_1	C_1	C_2	C_2	C_3	C_3	C_3	C_3	C_4
Clustering $\mathcal{C}_2$	C_1	C_1	C_2	C_2	C_2	C_3	C_1	C_2	C_2	C_1	C_3

Information theory–based methods

A clustering assigns objects to clusters and thus can be regarded as a compression of the information carried by the objects. In other words, a clustering can be regarded as a compressed representation of a given set of objects. Therefore we can use information theory to compare a clustering and the ground truth as representations. This is the general idea behind the information theory–based methods.

For example, we can measure the amount of information needed to describe the ground truth given the distribution of a clustering output by a method. Better the clustering results approach the ground truth, less amount information is needed. This leads to a natural approach using conditional entropy.

Concretely, according to information theory, the entropy of a clustering $\mathcal{C}$ is

$$H(\mathcal{C}) = -\sum_{i=1}^{m} \frac{|C_i|}{n} \log \frac{|C_i|}{n},$$

and the entropy of the ground truth is

$$H(\mathcal{G}) = -\sum_{i=1}^{l} \frac{|G_i|}{n} \log \frac{|G_i|}{n}.$$

The conditional entropy of $\mathcal{G}$ given cluster C_i is

$$H(\mathcal{G}|C_i) = -\sum_{j=1}^{l} \frac{|C_i \cap G_j|}{|C_i|} \log \frac{|C_i \cap G_j|}{|C_i|}.$$

The conditional entropy of $\mathcal{G}$ given clustering $\mathcal{C}$ is

$$H(\mathcal{G}|\mathcal{C}) = \sum_{i=1}^{m} \frac{|C_i|}{n} H(\mathcal{G}|C_i) = -\sum_{i=1}^{m} \sum_{j=1}^{l} \frac{|C_i \cap G_j|}{n} \log \frac{|C_i \cap G_j|}{|C_i|}.$$

In addition to the simple conditional entropy, more sophisticated information theory–based measures may be used, such as normalized mutual information and variation of information.

Taking the case in Table 8.1 as an example, we can calculate

$$H(\mathcal{G}|\mathcal{C}_1) = -(\frac{4}{11} \log \frac{4}{4} + \frac{2}{11} \log \frac{2}{2} + \frac{4}{11} \log \frac{4}{4} + \frac{1}{11} \log \frac{1}{1}) = 0$$

and

$$H(\mathcal{G}|\mathcal{C}_2) = -(\frac{2}{11}\log\frac{2}{4} + \frac{2}{11}\log\frac{2}{4} + \frac{3}{11}\log\frac{3}{5} + \frac{2}{11}\log\frac{2}{5} + \frac{1}{11}\log\frac{1}{2} + \frac{1}{11}\log\frac{1}{2})$$
$$= 0.297.$$

Clustering $\mathcal{C}_1$ has better quality than $\mathcal{C}_2$ in terms of conditional entropy. Again, although $H(\mathcal{G}|\mathcal{C}_1) = 0$, conditional entropy cannot detect the issue that $\mathcal{C}_1$ splits the objects in G_1 into two clusters.

Pairwise comparison–based methods

The pairwise comparison–based methods treat each group in the ground truth as a class. For each pair of objects $o_i, o_j \in D$ $(1 \leq i, j \leq n, i \neq j)$, if they are assigned to the same cluster/group, the assignment is regarded as positive, and otherwise, negative. Then, depending on assignments of o_i and o_j into clusters $C(o_i)$, $C(o_j)$, $G(o_i)$, and $G(o_j)$, we have four possible cases.

	$C(o_i) = C(o_j)$	$C(o_i) \neq C(o_j)$
$G(o_i) = G(o_j)$	true positive	false negative
$G(o_i) \neq G(o_j)$	false positive	true negative

Using the statistics on pairwise comparison, we can assess the quality of the clustering results approaching the ground truth. For example, we can use the Jaccard coefficient, which is defined as

$$J = \frac{\text{true positive}}{\text{true positive} + \text{false negative} + \text{false positive}}.$$

Many other measures can be built based on the pairwise comparison statistics, such as Rand statistic, fowlkes-Mallows measure, BCubed precision, and recall. The pairwise comparison results can be further used to conduct correlation analysis. For example, we can form a binary matrix $\boldsymbol{G}$ according to the ground truth, where element $v_{ij} = 1$ if $G(o_i) = G(o_j)$, and otherwise 0. A binary matrix $\boldsymbol{C}$ can also be constructed in a similar way based on a clustering $\mathcal{C}$. We can analyze the element-wise correlation between the two matrixes and use the correlation to measure the quality of the clustering results. Clearly, the more correlated the two matrixes, the better the clustering results.

8.5.4 Intrinsic methods

When the ground truth of a data set is not available, we have to use an intrinsic method to assess the clustering quality. Unable to reference any external supervision information, the intrinsic methods have to come back to the fundamental intuition in clustering analysis, that is, examining how compact clusters are and how well clusters are separated. Many intrinsic methods take the advantage of a similarity or distance measure between objects in the data set.

For example, the **Dunn index** measures the compactness of clusters by the maximum distance between two points that belong to the same cluster, that is, $\Delta = \max_{C(o_i)=C(o_j)}\{d(o_i, o_j)\}\}$. It measures the degree of separation among different clusters by the minimum distance between two points that belong to different clusters, that is $\delta = \min_{C(o_i)\neq C(o_j)}\{d(o_i, o_j)\}$. Then, the Dunn index is simply the ration $DI = \frac{\delta}{\Delta}$. The larger the ratio, the farther away the clusters are separated comparing to the compactness of the clusters.

The Dunn index uses the extreme distances to measure the cluster compactness and intercluster separation. The measures δ and Δ may be affected by the outliers. Many methods consider the average situations. The **silhouette coefficient** is such a measure. For a data set, D, of n objects, suppose D is partitioned into k clusters, $C_1, \ldots, C_k$. For each object $\boldsymbol{o} \in D$, we calculate $a(\boldsymbol{o})$ as the average distance between $\boldsymbol{o}$ and all other objects in the cluster to which $\boldsymbol{o}$ belongs. Similarly, $b(\boldsymbol{o})$ is the minimum average distance from $\boldsymbol{o}$ to all clusters to which $\boldsymbol{o}$ does not belong. Formally, suppose $\boldsymbol{o} \in C_i$ $(1 \leq i \leq k)$. Then,

$$a(\boldsymbol{o}) = \frac{\sum_{\boldsymbol{o}' \in C_i, \boldsymbol{o} \neq \boldsymbol{o}'} dist(\boldsymbol{o}, \boldsymbol{o}')}{|C_i| - 1} \tag{8.30}$$

and

$$b(\boldsymbol{o}) = \min_{C_j : 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{\boldsymbol{o}' \in C_j} dist(\boldsymbol{o}, \boldsymbol{o}')}{|C_j|} \right\}. \tag{8.31}$$

The **silhouette coefficient** of $\boldsymbol{o}$ is then defined as

$$s(\boldsymbol{o}) = \frac{b(\boldsymbol{o}) - a(\boldsymbol{o})}{\max\{a(\boldsymbol{o}), b(\boldsymbol{o})\}}. \tag{8.32}$$

The value of the silhouette coefficient is between -1 and 1. The value of $a(\boldsymbol{o})$ reflects the compactness of the cluster to which $\boldsymbol{o}$ belongs. The smaller the value, the more compact the cluster. The value of $b(\boldsymbol{o})$ captures the degree to which $\boldsymbol{o}$ is separated from other clusters. The larger $b(\boldsymbol{o})$ is, the more separated $\boldsymbol{o}$ is from other clusters. Therefore when the silhouette coefficient value of $\boldsymbol{o}$ approaches 1, the cluster containing $\boldsymbol{o}$ is compact and $\boldsymbol{o}$ is far away from other clusters, which is the preferable case. However, when the silhouette coefficient value is negative (i.e., $b(\boldsymbol{o}) < a(\boldsymbol{o})$, this means that, in expectation, $\boldsymbol{o}$ is closer to the objects in another cluster than to the objects in the same cluster as $\boldsymbol{o}$. In many cases, this is a bad situation and should be avoided.

To measure a cluster's fitness within a clustering, we can compute the average silhouette coefficient value of all objects in the cluster. To measure the quality of a clustering, we can use the average silhouette coefficient value of all objects in the data set. The silhouette coefficient and other intrinsic measures can also be used in the elbow method to heuristically derive the number of clusters in a data set by replacing the sum of within-cluster variances.

8.6 Summary

- A **cluster** is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters. The process of grouping a set of physical or abstract objects into classes of *similar* objects is called **clustering**.
- Cluster analysis has extensive **applications**, including business intelligence, image pattern recognition, Web search, biology, and security. Cluster analysis can be used as a standalone data mining tool to gain insight into the data distribution or as a preprocessing step for other data mining algorithms operating on the detected clusters.
- Clustering is a dynamic field of research in data mining. It is related to **unsupervised learning** in machine learning.