**What Is Cluster Analysis?**

Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. In this context, different clustering methods may generate different clusterings on the same data set. The partitioning is not performed by humans, but by the clustering algorithm. Hence, clustering is useful in that it can lead to the discovery of previously unknown groups within the data. Cluster analysis has been widely used in many applications such as business intelligence, image pattern recognition, Web search, biology, and security. In business intelligence, clustering can be used to organize a large number of customers into groups, where customers within a group share strong similar characteristics. This facilitates the development of business strategies for enhanced customer relationship management. Moreover, consider a consultant company with a large number of projects. To improve project management, clustering can be applied to partition projects into categories based on similarity so that project auditing and diagnosis (to improve project delivery and outcomes) can be conducted effectively

The following are typical requirements of clustering in data mining.

**Scalability:** Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions or even billions of objects, particularly in Web search scenarios. Clustering on only a sample of a given large data set may lead to biased results. Therefore, highly scalable clustering algorithms are needed.

**Ability to deal with different types of attributes**: Many algorithms are designed to cluster numeric (interval-based) data. However, applications may require clustering other data types, such as binary, nominal (categorical), and ordinal data, or mixtures of these data types. Recently, more and more applications need clustering techniques for complex data types such as graphs, sequences, images, and documents.

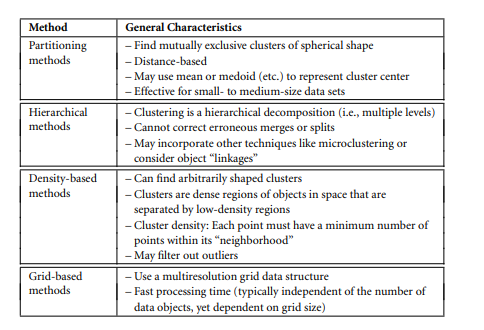
**Discovery of clusters with arbitrary shape**: Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures (Chapter 2). Algorithms based on such distance measures tend to find spherical clusters with similar size and density. However, a cluster could be of any shape. Consider sensors, for example, which are often deployed for environment surveillance. Cluster analysis on sensor readings can detect interesting phenomena. We may want to use clustering to find the frontier of a running forest fire, which is often not spherical. It is important to develop algorithms that can detect clusters of arbitrary shape.

**Requirements for domain knowledge to determine input parameters**: Many clustering algorithms require users to provide domain knowledge in the form of input parameters such as the desired number of clusters. Consequently, the clustering results may be sensitive to such parameters. Parameters are often hard to determine, especially for high-dimensionality data sets and where users have yet to grasp a deep understanding of their data. Requiring the specification of domain knowledge not only burdens users, but also makes the quality of clustering difficult to control.

**Ability to deal with noisy data**: Most real-world data sets contain outliers and/or missing, unknown, or erroneous data. Sensor readings, for example, are often noisy—some readings may be inaccurate due to the sensing mechanisms, and some readings may be erroneous due to interferences from surrounding transient objects. Clustering algorithms can be sensitive to such noise and may produce poor-quality clusters. Therefore, we need clustering methods that are robust to noise.

**Incremental clustering and insensitivity to input order**: In many applications, incremental updates (representing newer data) may arrive at any time. Some clustering algorithms cannot incorporate incremental updates into existing clustering structures and, instead, have to recompute a new clustering from scratch. Clustering algorithms may also be sensitive to the input data order. That is, given a set of data objects, clustering algorithms may return dramatically different clusterings depending on the order in which the objects are presented. Incremental clustering algorithms and algorithms that are insensitive to the input order are needed.

clustering method



Partitioning Methods

Formally, given a data set, D, of n objects, and k, the number of clusters to form, a partitioning algorithm organizes the objects into k partitions (k ≤ n), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are “similar” to one another and “dissimilar” to objects in other clusters in terms of the data set attributes

**k-Means: A Centroid-Based Technique**

Suppose a data set, D, contains n objects in Euclidean space. Partitioning methods distribute the objects in D into k clusters, C1,...,Ck , that is, Ci ⊂ D and Ci ∩Cj = ∅ for (1 ≤ i,j ≤ k). An objective function is used to assess the partitioning quality so that objects within a cluster are similar to one another but dissimilar to objects in other clusters. This is, the objective function aims for high intracluster similarity and low intercluster similarity. A centroid-based partitioning technique uses the centroid of a cluster, Ci , to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean or medoid of the objects (or points) assigned to the cluster. The difference between an object p ∈ Ci and ci , the representative of the cluster, is measured by dist(p,ci), where dist(x,y) is the Euclidean distance between two points x and y. The quality of cluster Ci can be measured by the withincluster variation, which is the sum of squared error between all objects in Ci and the centroid ci , defined as



“How does the k-means algorithm work?” The k-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster. It proceeds as follows. First, it randomly selects k of the objects in D, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the object and the cluster mean. The k-means algorithm then iteratively improves the within-cluster variation. For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration. All the objects are then reassigned using the updated means as the new cluster centers

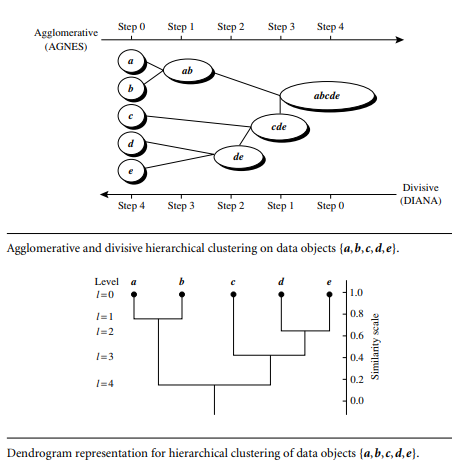
**k-Medoids: A Representative Object-Based Technique** The k-means algorithm is sensitive to outliers because such objects are far away from the majority of the data, and thus, when assigned to a cluster, they can dramatically distort the mean value of the cluster. This inadvertently affects the assignment of other objects to clusters

**(K-mean and K-medoid from class notes)**

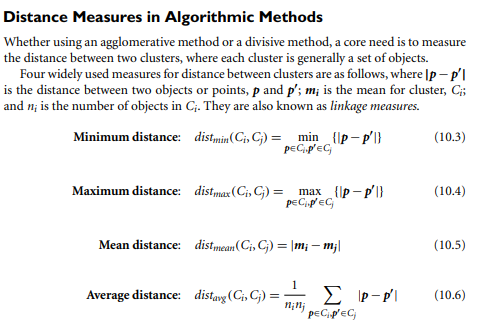
**A hierarchical clustering method works by grouping data objects into a hierarchy or “tree” of clusters**

**Agglomerative versus Divisive Hierarchical Clustering**

**A** hierarchical clustering method can be either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or topdown (splitting) fashion. Let’s have a closer look at these strategies. An agglomerative hierarchical clustering method uses a bottom-up strategy. It typically starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied. The single cluster becomes the hierarchy’s root. For the merging step, it finds the two clusters that are closest to each other (according to some similarity measure), and combines the two to form one cluster. Because two clusters are merged per iteration, where each cluster contains at least one object, an agglomerative method requires at most n iterations. A divisive hierarchical clustering method employs a top-down strategy. It starts by placing all objects in one cluster, which is the hierarchy’s root. It then divides the root cluster into several smaller subclusters, and recursively partitions those clusters into smaller ones. The partitioning process continues until each cluster at the lowest level is coherent enough—either containing only one object, or the objects within a cluster are sufficiently similar to each other. In either agglomerative or divisive hierarchical clustering, a user can specify the desired number of clusters as a termination condition.



This is a single-linkage approach in that each cluster is represented by all the objects in the cluster, and the similarity between two clusters is measured by the similarity of the closest pair of data points belonging to different clusters. The cluster-merging process repeats until all the objects are eventually merged to form one cluster. DIANA, the divisive method, proceeds in the contrasting way. All the objects are used to form one initial cluster. The cluster is split according to some principle such as the maximum Euclidean distance between the closest neighboring objects in the cluster. The cluster-splitting process repeats until, eventually, each new cluster contains only a single object. A tree structure called a dendrogram is commonly used to represent the process of hierarchical clustering. It shows how objects are grouped together (in an agglomerative method) or partitioned (in a divisive method) step-by-step

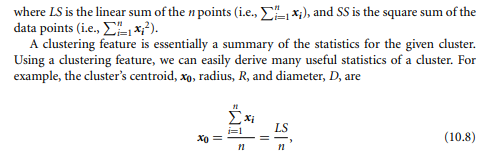


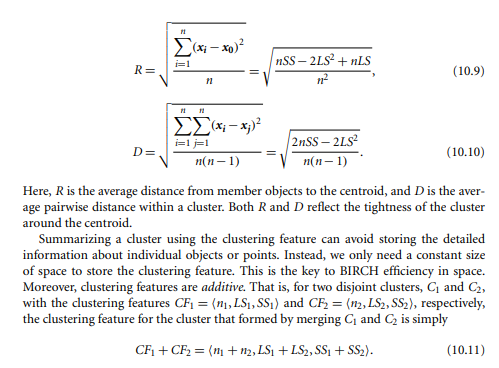
When an algorithm uses the minimum distance, dmin(Ci ,Cj), to measure the distance between clusters, it is sometimes called a nearest-neighbor clustering algorithm. Moreover, if the clustering process is terminated when the distance between nearest clusters exceeds a user-defined threshold, it is called a single-linkage algorithm. If we view the data points as nodes of a graph, with edges forming a path between the nodes in a cluster, then the merging of two clusters, Ci and Cj , corresponds to adding an edge between the nearest pair of nodes in Ci and Cj . Because edges linking clusters always go between distinct clusters, the resulting graph will generate a tree.

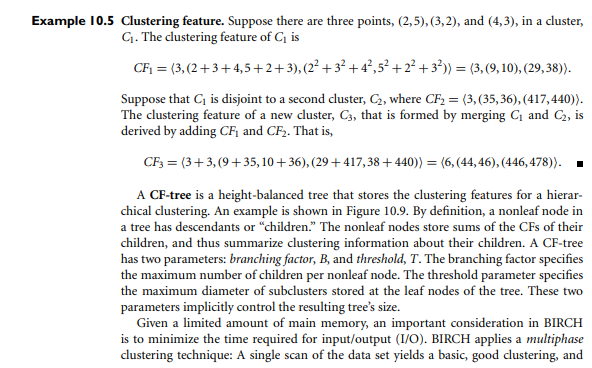
**BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees**

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is designed for clustering a large amount of numeric data by integrating hierarchical clustering (at the initial microclustering stage) and other clustering methods such as iterative partitioning (at the later macroclustering stage). It overcomes the two difficulties in agglomerative clustering methods: (1) scalability and (2) the inability to undo what was done in the previous step. BIRCH uses the notions of clustering feature to summarize a cluster, and clustering feature tree (CF-tree) to represent a cluster hierarchy.

The clustering feature (CF) of the cluster is a 3-D vector summarizing information about clusters of objects. It is defined as CF = <n,LS,SSi>





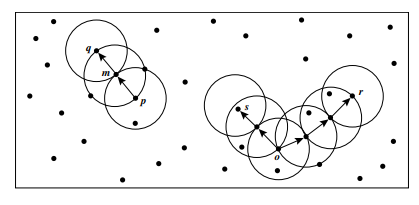


one or more additional scans can optionally be used to further improve the quality. The primary phases are Phase 1: BIRCH scans the database to build an initial in-memory CF-tree, which can be viewed as a multilevel compression of the data that tries to preserve the data’s inherent clustering structure. Phase 2: BIRCH applies a (selected) clustering algorithm to cluster the leaf nodes of the CF-tree, which removes sparse clusters as outliers and groups dense clusters into larger ones.

**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 o can be measured by the number of objects close to 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. “How does DBSCAN quantify the neighborhood of an object?” A user-specified parameter > 0 is used to specify the radius of a neighborhood we consider for every object. The -neighborhood of an object o is the space within a radius centered at o. Due to the fixed neighborhood size parameterized by , 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 use parameter, MinPts, which specifies the density threshold of dense regions. An object is a core object if the -neighborhood of the object contains at least MinPts objects. Core objects are the pillars of dense regions

“How can we assemble a large dense region using small dense regions centered by core objects?” In DBSCAN, p is density-reachable from q (with respect to and MinPts in D) if there is a chain of objects p1,...,pn, such that p1 = q, pn = p, and pi+1 is directly density-reachable from pi with respect to and MinPts, for 1 ≤ i ≤ n, pi ∈ D. Note that density-reachability is not an equivalence relation because it is not symmetric. If both o1 and o2 are core objects and o1 is density-reachable from o2, then o2 is density-reachable from o1. However, if o2 is a core object but o1 is not, then o1 may be density-reachable from o2, but not vice versa. To connect core objects as well as their neighbors in a dense region, DBSCAN uses the notion of density-connectedness. Two objects p1,p2 ∈ D are density-connected with respect to and MinPts if there is an object q ∈ D such that both p1 and p2 are densityreachable from q with respect to and MinPts. Unlike density-reachability, densityconnectedness is an equivalence relation. It is easy to show that, for objects o1, o2, and o3, if o1 and o2 are density-connected, and o2 and o3 are density-connected, then so are o1 and o3.

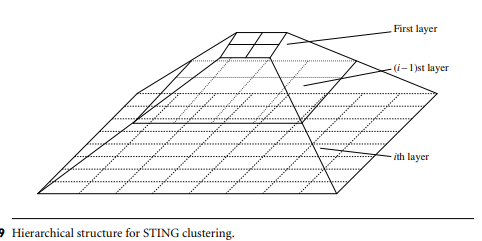
\

“How does DBSCAN find clusters?” Initially, all objects in a given data set D are marked as “unvisited.” DBSCAN randomly selects an unvisited object p, marks p as “visited,” and checks whether the -neighborhood of p contains at least MinPts objects. If not, p is marked as a noise point. Otherwise, a new cluster C is created for p, and all the objects in the -neighborhood of p are added to a candidate set, N. DBSCAN iteratively adds to C those objects in N that do not belong to any cluster. In this process, for an object p 0 in N that carries the label “unvisited,” DBSCAN marks it as “visited” and checks its -neighborhood. If the -neighborhood of p 0 has at least MinPts objects, those objects in the -neighborhood of p 0 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.

**Grid-Based Methods**

The clustering methods discussed so far are data-driven—they partition the set of objects and adapt to the distribution of the objects in the embedding space. Alternatively, a grid-based clustering method takes a space-driven approach by partitioning the embedding space into cells independent of the distribution of the input objects. 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

**STING: STatistical INformation Grid STING** is a grid-based multi resolution clustering technique in which the embedding 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. Figure 10.19 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. 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



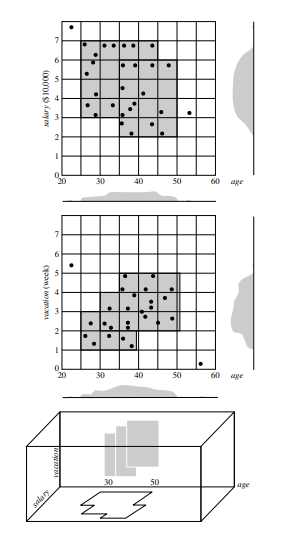
The type of distribution of a higher-level cell can be computed based on the majority of distribution types of its corresponding lower-level cells in conjunction with a threshold filtering process. If the distributions of the lower-level cells disagree with each other and fail the threshold test, the distribution type of the high-level cell is set to none.

**“What advantages does STING offer over other clustering methods?”** STING offers several advantages: (1) 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; (2) the grid structure facilitates parallel processing and incremental updating; and (3) the method’s efficiency 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

**CLIQUE: An Apriori-like Subspace Clustering Method**

A data object often has tens of attributes, many of which may be irrelevant. The values of attributes may vary considerably. These factors can make it difficult to locate clusters that span the entire data space. It may be more meaningful to instead search for clusters within different subspaces of the data. For example, consider a healthinformatics application where patient records contain extensive attributes describing personal information, numerous symptoms, conditions, and family history. Finding a nontrivial group of patients for which all or even most of the attributes strongly agree is unlikely. In bird flu patients, for instance, the age, gender, and job attributes may vary dramatically within a wide range of values. Thus, it can be difficult to find such a cluster within the entire data space. Instead, by searching in subspaces, we may find a cluster of similar patients in a lower-dimensional space (e.g., patients who are similar to one other with respect to symptoms like high fever, cough but no runny nose, and aged between 3 and 16). CLIQUE (CLustering In QUEst) is a simple grid-based method for finding densitybased clusters in subspaces. CLIQUE partitions each dimension into nonoverlapping intervals, thereby partitioning the entire embedding space of the data objects 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 6). 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 Figure 10.20, where the embedding data space contains three dimensions: age, salary, and vacation. A 2-D 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 point



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, c1 and c2, in subspaces (Di1 ,...,Dik ) and (Dj1 ,...,Djk ), respectively, if Di1 = Dj1 , . . . , Dik−1 = Djk−1 , and c1 and c2 share the same intervals in those dimensions. The join operation generates a new (k + 1)-dimensional candidate cell c in space (Di1 ,...,Dik−1 ,Dik ,Djk ). 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 second 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 8) 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.

**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”?

**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**

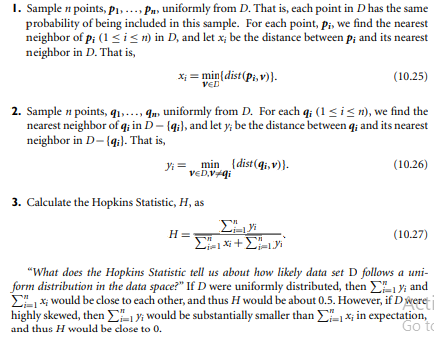
A few algorithms, such as k-means, require the number of clusters in a data set as the parameter. Moreover, the number of clusters can be regarded as an interesting and important summary statistic of a data set. Therefore, it is desirable to estimate this number even before a clustering algorithm is 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

**Assessing Clustering Tendency**

“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’s 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 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:



**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. On the other hand, 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 best cost. However, having one object per cluster does not enable any data summarization.

A simple method is to set the number of clusters to about q n 2 for a data set of n points. In expectation, each cluster has √ 2n points.

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.

**Measuring Clustering Quality**

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?” 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 extrinsic methods, which compare the clustering against the group truth and measure. If the ground truth is unavailable, we can use 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, while intrinsic methods are unsupervised methods

**Extrinsic Methods**

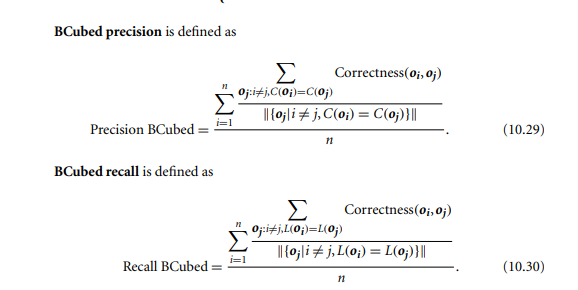
**Cluster homogeneity**. This requires that the more pure the clusters in a clustering are, the better the clustering. Suppose that ground truth says that the objects in a data set, D, can belong to categories L1,...,Ln. Consider clustering, C1, wherein a cluster C ∈ C1 contains objects from two categories Li ,Lj (1 ≤ i < j ≤ n). Also consider clustering C2, which is identical to C1 except that C2 is split into two clusters containing the objects in Li and Lj , respectively. A clustering quality measure, Q, respecting cluster homogeneity should give a higher score to C2 than C1, that is, Q(C2,Cg ) > Q(C1,Cg ).

**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 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 ground truth) to the same cluster. Consider clustering C1, which contains clusters C1 and C2, of which the members belong to the same category according to ground truth. Let clustering C2 be identical to C1 except that C1 and C2 are merged into one cluster in C2. Then, a clustering quality measure, Q, respecting cluster completeness should give a higher score to C2, that is, Q(C2,Cg ) > Q(C1,Cg ).

**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 C1 and a cluster C ∈ C1 such that all objects in C except for one, denoted by o, belong to the same category according to ground truth. Consider a clustering C2 identical to C1 except that o is assigned to a cluster C 0 6= C in C2 such that C 0 contains objects from various categories according to ground truth, and thus is noisy. In other words, C 0 in C2 is a rag bag. Then, a clustering quality measure Q respecting the rag bag criterion should give a higher score to C2, that is, Q(C2,Cg ) > Q(C1,Cg ).

**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 o1, . . . , on, belong to one category and the other two objects, denoted by on+1,on+2, belong to another category. Suppose clustering C1 has three clusters, C1 = {o1, . . . , on}, C2 = {on+1}, and C3 = {on+2}. Let clustering C2 have three clusters, too, namely C1 = {o1, . . . , on−1}, C2 = {on}, and C3 = {on+1,on+2}. In other words, C1 splits the small category and C2 splits the big category. A clustering quality measure Q preserving small clusters should give a higher score to C2, that is, Q(C2,Cg ) > Q(C1,Cg ).

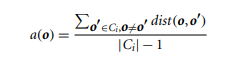
BCubed evaluates the precision and recall for every object in a clustering on a given data set according to ground truth. The precision of an object indicates how many other objects in the same cluster belong to the same category as the object. The recall of an object reflects how many objects of the same category are assigned to the same cluster

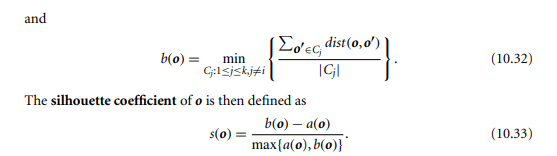


**Intrinsic Methods**

When the ground truth of a data set is not available, we have to use an intrinsic method to assess the clustering quality.

In general, intrinsic methods evaluate a clustering by examining how well the clusters are separated and how compact the clusters are. Many intrinsic methods have the advantage of a similarity metric between objects in the data set. The silhouette coefficient is such a measure. For a data set, D, of n objects, suppose D is partitioned into k clusters, C1,...,Ck . For each object o ∈ D, we calculate a(o) as the average distance between o and all other objects in the cluster to which o belongs. Similarly, b(o) is the minimum average distance from o to all clusters to which o does not belong. Formally, suppose o ∈ Ci (1 ≤ i ≤ k); then





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