Chapter 5

Overview on Techniques in Cluster Analysis

Itziar Frades and Rune Matthiesen

Abstract

Clustering is the unsupervised, semisupervised, and supervised classification of patterns into groups. The clustering problem has been addressed in many contexts and disciplines. Cluster analysis encompasses different methods and algorithms for grouping objects of similar kinds into respective categories. In this chapter, we describe a number of methods and algorithms for cluster analysis in a stepwise framework. The steps of a typical clustering analysis process include sequentially pattern representation, the choice of the similarity measure, the choice of the clustering algorithm, the assessment of the output, and the representation of the clusters.

Key words: Clustering algorithm, feature selection, feature extraction, similarity measure, cluster tendency, cluster validity, cluster stability, relevance networks, dendrogram.

1. Introduction

1.1. The Importance of Clustering

Clustering is one of the most useful tasks in the data mining process for discovering groups and identifying new interesting patterns in the underlying data. Clustering algorithms partition data objects into subsets (clusters) based on similarity or dissimilarity. Patterns within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster. The clustering process is an unsupervised, semisupervised, or supervised method. Since unsupervised cluster algorithms do not use predefined class labels or examples that would indicate grouping properties in the data set, it is the ideal method for identifying new patterns in data. Unsupervised clustering is also frequently used in combination with other supervised classification algorithms since it has the potential to detect incorrect class labels, outliers, errors, bias, and bad experimental designs.

R. Matthiesen (ed.), *Bioinformatics Methods in Clinical Research*, Methods in Molecular Biology 593,
DOI 10.1007/978-1-60327-194-3_5, © Humana Press, a part of Springer Science+Business Media, LLC 2010

Table 5.1
Overview of the methods discussed in the chapter

Pattern representation	Similarity measure	Clustering algorithm	Assessment of the output	Representation of clusters
Feature selection (1)	Euclidean distance (2)	*Hierarchical* BIRCH (3), CURE (4), ROCK (5), DIANA (6), MONA (6),	*Clustering tendency* (7, 8)	*Graphs* Relevance networks (9)
	Manhattan distance (10)	*Partitional* *k*-means (11), ISODATA (12), PAM (6), CLARA (6), CLARANS (13), nearest neighbor (14)		*Partitions* (15)
	Pearson's correlation coefficient (16)	*Density-based* DBSCAN(17), DENCLUE (18)	*Cluster validity* External, internal, and relative criteria. Validity (19) indices (Dunn's) (20)	*Classification trees* (15)
	Vector angle distance (21)	*Grid-based* WaveCluster (22) and STING (23)		
Feature extraction (Principal component analysis) (24)	Squared Pearson's correlation (16)	*Fuzzy clustering* FCM (25)	*Cluster stability* Bagging (26)	*Dendrogram* Displaying the assessment of the uncertainty in hierarchical cluster analysis (27)
	Inner product (28)	Artificial neural networks for clustering. SOM (29), SOTA (30, 31)		
	Spearman's rank correlation (32) and Kendall's Tau (33)	*Evolutionary approaches for clustering* Genetic algorithms (34)		
	Mutual information (35)	Biclustering (36)		

There are an overwhelming number of different strategies that can be combined in cluster analysis. In the following sections, an attempt to discuss a subset of the available clustering methods is provided (*see* **Table 5.1**).

1.2. Computational Steps in the Clustering Process

A typical clustering analysis can be subdivided into the steps outlined below (9). The actual choices made in each step are data-dependent and in many cases are optimized by trial and error.

1. **Pattern representation** (optionally including feature extraction and/or selection): The goal is to select the features on which the clustering is to be performed. The features should encode as much information as possible.
2. **Similarity measure:** Definition of a pattern proximity measure appropriate to the data domain. The similarity measure quantifies how similar two data points or patterns are. In most cases, it is important to check that all selected features contribute equally to the computation of the proximity measure and that no features dominate others.
3. **Clustering algorithm**: This step refers to the choice of the clustering algorithm. It should result in the definition of a good clustering scheme for the data set under analysis. The clustering algorithm is also critical for computational speed.
4. **Assessment of the output:** Clustering algorithms define clusters that are unknown a priori; therefore, the final partition of data requires some kind of evaluation. To verify whether the result of a clustering algorithm is correct, appropriate criteria and techniques must be used. These techniques aim at the quantitative evaluation of the results of the clustering algorithms and are known under the general term of *cluster validity* methods. They answer questions like "how many clusters are there in the data set?", "does the resulting clustering scheme fits our data set?", "is there a better partitioning for our data set?", and "how consistent or robust are the clusters when re-sampling the data?"
5. **Graphical representation:** The cluster results need to be represented with some sort of data abstraction in a graphical display for easy interpretation.

Each of the above steps is discussed in more detail in the sections below.

2. Pattern Representation

A *pattern* (or *feature vector*, *observation*, *object,* or *data point*) (*see* **Note 1**) is a single data item used by the clustering algorithm. It typically consists of a vector of d measurements:

x = (*x*1, . . .,*xd*). The individual scalar components *xi* of a pattern are called *features* (or *attributes*). Pattern representation is concerned with the preprocessing of feature extraction and feature selection and is used to define the number of features, type, weight, and scale of the features available to the clustering algorithm. For example, in mass spectrometry, one can choose to use all data points in a spectrum or only the data points corresponding to peak tops that will affect the number of features. The peak tops can be represented as a signal-to-noise ratio, integrated intensity, or maximum intensity.

The practice of assigning different weights to features and/or scaling of their values is widespread. Translating the importance of each feature using weights allows clusters to be constructed of better shapes and can lead to better classification results (37). Attribute scaling is the application of a mathematical transformation to each of the individual components of the attributes so that all of the attributes make a comparable contribution to the measurement of similarity.

2.1. Feature Selection

Feature selection is the process of identifying the most effective subset of the original features to use in the clustering. It implies selecting a subset of the existing features without any transformation. It is important to distinguish feature selection from dimension reduction. In feature selection, some features are completely removed, whereas in dimension reduction, new features are defined as functions of all features. The problem of feature selection is defined as follows: Given a set of features, select a subset that leads to the smallest clustering error performance. Using all features in a data set often introduces noise, which can confuse the clustering algorithm. Feature selection removes noisy features, improves the performance of the clustering algorithms, increases the speed of the clustering algorithms, and yields a more compact, more easily interpretable representation of the target concept (38).

Search and evaluation of subsets of features are the two main steps in the feature selection process. Search methods can be exhaustive, heuristic, random, or some hybrid of these techniques. Their efficiency is measured by optimality, defined as the best subset of features according to a specified criterion. Exhaustive methods guarantee optimality but are impractical due to their exponential time complexity in a number of features. Random methods generate subsets randomly and return the best subset at any point of time, approaching optimality only asymptotically. A variation of pure random methods is the probabilistic method where the probability of generating a subset varies by some rules. Examples of such rules are genetic algorithms and simulated annealing. Forward and backward selection are examples of heuristic methods. A forward selection method first finds the best feature among all features and stepwise accumulates the features until the performance drops. A backward selection algorithm is the opposite of the forward selection algorithm (39).

When selecting a good attribute subset, there are two fundamentally different approaches. One is to make an independent assessment based on general characteristics of the data; the other is to evaluate the subset using the machine learning algorithm that will ultimately be employed for learning. The latter approach requires that the class labels are known. The first is called a *filter* method, because the attribute set is filtered to produce the most promising subset before the learning commences (39, 40). The second is known as a *wrapper* method, because the learning algorithm is wrapped into the selection procedure (41, 42). Wrapper methods typically require extensive computation to search the best features (38).

The relevance of the features can be evaluated either individually (univariate approaches) or in a multivariate manner. Univariate approaches are simple and fast; however, possible correlation and dependencies between the features are not considered (1).

2.2. Feature Extraction

Feature extraction is the process of using transformations of the input features to produce a new salient feature set. It involves transforming the existing features into a lower-dimensional space. Linear transforms, such as principal component analysis (24), factor analysis (43), linear discriminant analysis (44), partial least-squares regression (45), and projection pursuit (46), have been widely used in feature extraction and dimensionality reduction. The most commonly used linear feature extractor is the principal components analysis.

2.2.1. Principal Component Analysis

Principal component analysis (PCA) (24) is a technique used to reduce multidimensional data sets to lower dimensions for analysis. PCA is an unsupervised technique and as such does not include label or category information given in the data.

In general, PCA changes the original variables into new independent and uncorrelated variables called *principal components* that explain the observed variability. PCA performs an orthogonal linear transformation. This transforms the data into a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate, the second-greatest variance on the second coordinate, and so on. These coordinates are the principal components. In other words, given m observations on n variables, the goal of PCA is to reduce the dimensionality of the data matrix by finding $r \leq n$ new variables. These r principal components account together for as much of the variance in the original n variables as possible while remaining mutually uncorrelated and orthogonal. For each component, it is possible to find one eigenvalue with an associated variance value. The eigenvalues and their corresponding eigenvectors originate from the covariance matrix obtained from the original data.

The result is a lower-dimensional representation of the data that removes some of the "noisy" directions, making the data more accessible for visualization and analysis. The problem with using PCA is that it assumes that the large variance between groups is the most important, which is not necessarily always true. For example, small changes in the concentration of electrolytes in the blood can have a profound effect on the health of an individual. In such cases, an alternative to PCA is partial least-squares regression (PLS regression) (47). In PLS regression, the overall goal is to extract some factors to predict responses in the population and to describe the common structure underlying the dependent and independent variables. PLS regression searches for a set of components (called *latent vectors*; *see* **Note 2**) that performs a simultaneous decomposition of X (data matrix from independent variables) and Y (data matrix from dependent variables) with the constraint that these components explain as much as possible of the covariance between X and Y. This first step is followed by a regression step where the decomposition of X (latent variables) is used to predict Y.

3. Similarity Measure

A metric system is a decimalized system of measurement (*see* **Note 3**). The similarity measure quantifies how similar two data points (or two objects) are and will provide an indication of proximity, likeness, affinity, or association. Because of the variety of feature types and scales, the similarity measure must be chosen carefully.

There are two types of similarity measures: metric and probability distribution–based similarity measures.

A metric system is a decimalized system of measurement (*see* **Note 3**). A number of the more common metrics are discussed below (15, 48); in each case, X_i and Y_i are two n-dimensional vectors (patterns) being compared.

Euclidean distance: The most popular metric for continuous features is the Euclidean distance (2) d_e:

$$d_e = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2} \qquad [1]$$

where the index i iterates over all values in the vectors. The Euclidian distance metric (2) is a measure of the geometric distance between two components. It is purely based on magnitude. So the case of two vectors whose values are clearly highly correlated would not be well represented by the Euclidian distance d_e, where only the distance between them is taken into account.

Manhattan distance: Manhattan distance (10) d_M is similar to the Euclidian distance. It is calculated as the sum of the absolute distances of two vector values (10):

$$d_M = \sum_{i=1}^{n} |X_i - Y_i| \qquad [2]$$

Pearson's correlation: Pearson's correlation coefficient (16) r is calculated by

$$r = \frac{\sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}} \qquad [3]$$

where $\overline{X}$ is the mean of vector X and $\overline{Y}$ is the mean of vector Y_i. Pearson's correlation coefficient is a measure of the tendency of the vectors to increase or decrease together; in other words, it measures the association between two variables. It ranges from –1 (negatively correlated), to 0 (no correlation), to 1 (positively correlated), with 1 meaning that the two series are identical, 0 meaning they are completely independent, and −1 meaning they are perfect opposites. It measures only linear correlations. The correlation coefficient is invariant under a scalar transformation of the data (adding, subtracting, or multiplying the vectors with a constant factor), meaning that it is independent of both origin and scale. If two patterns have a common peak or valley at a single feature, the correlation will be dominated by this feature, although the patterns at the remaining features may be completely dissimilar. As a consequence, it is very sensitive to outliers. Another drawback of Pearson's correlation coefficient is that it assumes an approximate Gaussian distribution of the points and may not be robust for non-Gaussian distributions (49).

Vector angle distance: Cos(α) (21), where 1 − Cos(α) is a measure of the angle between two vectors, is also known as the uncentered Pearson's correlation distance. It captures a similarity that does not change if scales are multiplied by a common factor. Another strong property is that the cosine similarity does not depend on the length of the vectors. The cosine measure assigns a high similarity to points that are in the same direction from the origin, zero similarity to points that are perpendicular to one another, and negative similarity for those that are pointing in opposing directions to one another. This is basically the same formula as above, except the mean is expected to be 0.

$$\cos\alpha = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}} \qquad [4]$$

Squared Pearson's correlation coefficient: The squared Pearson correlation coefficient (16) r_{sq} calculates the square of the Pearson correlation coefficient, so that negative values become positive. It is a measure of how well the regression line approximates the real data points. An r_{sq} of 1 indicates that the regression line perfectly fits the data.

$$r_{sq} = \left(\frac{\sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}} \right)^2 \qquad [5]$$

Inner product: The simplest measurement of association between two vectors is the inner product, also referred as the scalar or the dot product (28). It is a value expressing the angular relationship between two vectors. When the two vectors are perpendicular, the result of the inner product will be zero, because the cosine of 90° will be zero. If the angle between the two vectors is less than 90°, the dot product will be positive as the cosine will be positive, and the vector lengths are always positive values. If the angle between the two vectors is greater than 90°, the dot product will be negative, as the cosine will be negative, and the vector lengths are always positive values. The inner product between X and Y is defined as the sum of products of components and can be modified by defining an adjusted or averaged dot product d:

$$d = \frac{1}{n}\sum_{i=1}^{n} X_i Y_i = \frac{1}{n}\sum_{i=1}^{n} |X_i|\,|Y_i| \cos\alpha \qquad [6]$$

Rank-based metrics: Spearman's rank correlation (32) and Kendall's Tau (33) are nonparametric or rank correlations. They are techniques for determining the correlation between two ordinal variables (*see* **Note 4**) or metric variables reduced to an ordinal scale, and they are used to measure the degree of correspondence between the resulting two rankings. When replacing the value of each X_i and Y_i by the value of its rank among all the other, that is, 1, 2, 3, . . ., n, the resulting list of numbers will be drawn from a perfectly known distribution function, from the integers from 1 to n. Spearman's correlation coefficient and Kendall's Tau do not require the assumption of Gaussian distribution and therefore are more robust against outliers than Pearson's correlation coefficient. However, as a consequence of ranking, a significant amount of information present in the data is lost.

Mutual information (35) and the Kullback–Leibler divergence (50) are examples of probability distribution–based similarity measures. Mutual information is a special case of Kullback–Leibler divergence. In fact, many of the quantities in information theory can be considered as special cases of Kullback–Leibler divergence.

Kullback–Leibler divergence: For two probability functions P and Q of discrete random variables, the Kullback–Leibler divergence is defined by

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(I)}{Q(I)} \qquad [7]$$

Mutual information: Mutual information (35) is a quantity that measures the mutual or statistical dependence of the two variables. It quantifies the reduction in the uncertainty of one random variable given knowledge about another random variable. It takes into account nonlinear correlations. The mutual information of two discrete random variables X and Y, $I(X; Y)$, can be defined as

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \qquad [8]$$

where $p(x)$, $p(y)$, and $p(x, y)$ are thge probabilities of a given x- and y-value, and the co-occurrence of the x- and y-values. Mutual information can be equivalently expressed as

$$\begin{aligned} I(X;Y) &= H(X) - H(X/Y) = H(Y) - H(Y/X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \qquad [9]$$

$$H(X) = -\sum_{x\in X} P(x) \log(P(x)) \qquad [10]$$

$$H(Y) = -\sum_{y\in Y} P(y) \log(P(y)) \qquad [11]$$

$$H(X;Y) = -\sum_{x\in X, y\in Y} P(x,y) \log(P(x,y)) \qquad [12]$$

where $H(X)$ and $H(Y)$ are the marginal entropies, $H(X|Y)$ and $H(Y_i|X)$ are the conditional entropies, and $H(X,Y)$ is the joint entropy of X and Y. Recall that the entropy quantifies uncertainty (*see* **Note 5**).

4. Clustering Algorithm Categories

According to the method adopted to define clusters, the algorithms can be broadly classified into the categories described below (15, 19, 51):

- *Hierarchical clustering* proceeds iteratively by either merging smaller clusters into larger ones, or by splitting larger clusters, resulting in a tree of clusters, called a *dendrogram*, that shows how the clusters are related. Typical examples for hierarchical clustering include BIRCH (3), CURE (4), and ROCK (5).
- *Partitional clustering* decomposes the data set into a set of disjoint clusters. More specifically, it attempts to determine an integer number of partitions that optimize in an iterative way a certain criterion function that may represent the local or global structure of the data. The simplest and most commonly used partitional algorithm is *k*-means (11).
- *Density-based clustering* proceeds by grouping neighboring patterns of a data set into clusters based on density conditions. Some examples include DBSCAN (17) and DENCLUE (18).
- *Grid-based clustering* operates with the assumption that the space is divided into a finite number of cells and all of the operations are done in the divided space. STING (23) and WaveCluster (22) are examples of programs that implement this type of clustering.

  For each of above categories, there are a number of subtypes and different algorithms for finding the clusters (15).

Another classification criterion is the way clustering handles uncertainty in terms of cluster overlap. The output clustering can be hard (crisp) or fuzzy (soft): A hard clustering algorithm allocates each pattern to a single cluster. In a fuzzy clustering method, each pattern has a variable degree of membership in each of the output clusters. If an instance belongs to a group with a certain probability, the clustering is *probabilistic*; if not, it is said to be *deterministic*. Probabilistic clustering algorithms assume an underlying probability model with parameters that describe the probability that an instance belongs to a certain cluster.

Two schemes are related to the sequential or simultaneous use of features in the clustering process. In a monothetic scheme, cluster membership is based on the presence or absence of a single feature. Polythetic schemes use more than one feature.

5. Clustering Algorithms

5.1. Hierarchical Clustering Algorithms

Hierarchical clustering (52) transforms a distance matrix of pairwise similarity measurements between all items into a hierarchy of nested groupings. The hierarchy is represented with a binary tree-like dendrogram that shows the nested grouping of patterns and the similarity levels at which groupings change. According to the algorithmic structure and operation, hierarchical algorithms can be further categorized into two procedures:

1. Agglomerative procedures: This procedure begins with each pattern in a distinct cluster and successively merges clusters together until a stopping criterion is satisfied.
2. Divisive procedures: A divisive method begins with all patterns in a single cluster and iteratively splits the cluster until a stopping criterion is met.

The basic process of agglomerative hierarchical clustering has the following steps:

1. A similarity distance matrix is constructed by calculating the pairwise distance between all patterns. Each pattern is assigned to a single cluster, so each pattern represents one cluster.
2. The two clusters *r* and *s* with the minimum distance to each other are found.
3. The clusters *r* and *s* are merged and *r* is replaced with the new cluster. *s* is deleted and distances (similarities) between the new cluster and each of the old clusters are computed.
4. Repeat steps 2 and 3 until the total number of clusters is one.

In the first step, when each item represents its own cluster, the distances between those items are defined by the chosen similarity measure. Then a linkage or amalgamation rule is needed to determine if two clusters are sufficiently similar to be linked together (*see* **Note 6**). Step 3 can be done in different ways, which is what distinguishes *single-linkage* from *complete-linkage*, *average-linkage,* and *Ward's method* clustering:

In *single-linkage* clustering, the distance between one cluster and another is considered to be equal to the shortest distance from any member of one cluster to any member of the other cluster.

In *complete-linkage* clustering, the distances between clusters are determined by the greatest distance between any two members in the different clusters.

In *average-linkage* clustering, the distance between two clusters is calculated as the average distance between all pairs of members in the two different clusters.

Ward's method clustering (53) uses an analysis of a variance approach to evaluate the distances between clusters. This method attempts to minimize the sum of squares of any two clusters that can be formed at each step. In general, this method is very efficient; however, it tends to create clusters that are small in size.

Hierarchical clustering algorithms only require a matrix with the pairwise similarities based on a predefined distance. However, when the number of data points to be clustered, m, is large, such a matrix requires a lot of storage space, of the order $o(m^2)$. In recent years, because of the requirement for handling large-scale data sets, many new hierarchical clustering techniques have appeared and have greatly improved the clustering performance. Typical examples for agglomerative hierarchical clustering include BIRCH (3), CURE (4), and ROCK (5). Examples of divisive hierarchical clustering include DIANA (6) and MONA (6).

Hierarchical clustering is the most commonly used clustering strategy for gene expression analysis at the moment (48). The biggest advantage is that the only parameters that need to be specified are the type of similarity distance measurement and the choice of the amalgamation rule (*see* **Note 6**). However, if the feature selection procedure is used, many more parameters must be chosen and the choice of such parameters indeed becomes a parameter optimization problem. Hierarchical clustering algorithms are applicable to any attribute type, and they do not make any assumptions about the underlying data distribution. They do not require the number of clusters to be known in advance, while they provide a complete hierarchy of clusters, and a "flat" partition can be derived using a cut through the dendrogram. However, both divisive and agglomerative procedures suffer from their inability to perform adjustments once the splitting or merging decision is made. The deterministic nature of this procedure and the impossibility of reevaluating the results can cause the pattern to be clustered based on local decisions rather than a global one (54).

When clusters overlap or vary considerably in shape, density, or size, this class of methods is known to perform poorly (55). Hierarchical clustering presents drawbacks when dealing with data containing a high amount of noise and is dependent on the data order (30).

5.2. Partitional Clustering Algorithms

The result of a partitional clustering algorithm is a single partition of the data into a set of disjoint clusters. Partitional methods are used when the analysis involves very large data sets for which the construction of a dendrogram is computationally prohibitive. A drawback of partitional algorithms is that the number of clusters must be specified. In partitional techniques, the clusters produced optimize a criterion function defined either locally, that is, on a subset of patterns, or globally, so defined over all of the patterns. As the combinatorial search of the set of possible labeling for an optimum value of a criterion is computationally very expensive, in practice, the algorithm is run multiple times with different starting points, using as the output the best configuration obtained from all of the runs.

The most commonly used strategy in partitional clustering is based on the squared-error criterion. It tries to minimize the total intracluster variance, or the squared-error function:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} |x_j - \mu_i|^2 \qquad [13]$$

where there are k clusters S_i, $i = 1, 2, \ldots, k$, and μ_i is the centroid or mean point of all the points.

The strategy based on the squared-error criterion works well with isolated and compact clusters, but when outliers (*see* **Note 7**) are present, they can influence the clusters that are found because the resulting cluster centroids may not be representative.

5.2.1. k-Means

The simplest and most commonly used algorithm employing the squared-error criterion is the *k*-means (11) partitional clustering algorithm. The parameter *k,* which accounts for the number of clusters, must be specified. Then *k* points are chosen at random as cluster centers. All items are assigned to their closest cluster center according to the distance measure being used. Next the centroid, or mean, of the items in each cluster is calculated. These centroids are taken to be new center values for their respective clusters. The whole process is repeated with the new cluster centers. Iteration continues until the same or similar points are assigned to each cluster in consecutive rounds or there is a minimal decrease in the squared error. Using centroids has the advantage of a clear geometric and statistical meaning while keeping the algorithm insensitive to data ordering, but means of points within a cluster only work with numerical attributes and can be negatively affected by a single outlier (56).

The *k*-means is easy to implement and its time complexity is $O(1^* k^* n)$, where n is the number of patterns, l is the number of iterations, and k is the number of clusters.

The major drawback of this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function value if the initial partition is not properly chosen. To find the global optimum, techniques such as simulated annealing and generic algorithms can be incorporated with the *k*-means algorithm.

The basic steps of the *k*-means algorithm are as follows:

1. An initial partition with *k* clusters containing randomly chosen samples is selected, and the centroids of the clusters are computed.
2. By assigning each sample to the closest cluster center, a new partition is generated.
3. New cluster centers as the centroids of the clusters are recomputed.
4. Steps 2 and 3 are repeated until the optimal value of the criterion function is found or until the cluster membership stabilizes.

Some variants like the ISODATA (12) algorithm include a procedure to search for the best *k* and therefore overcome the problem of estimating *k*. ISODATA can dynamically adjust the number of clusters by merging and splitting clusters according to some predefined thresholds. In this sense, the problem of identifying the initial number of clusters becomes a matter of tweaking parameters.

5.2.2. Partitioning Around Medoids

Partitioning around medoids (PAM) (6) is a partitioning method that operates on a distance matrix and requires a prespecified number of clusters *k*. The PAM procedure is based on the search for *k* representative patterns, or medoids, among the observations. The medoids minimize the sum of the distances of the observations to their closest medoid. Therefore, a medoid can be defined as an object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal, resulting in the medoid's being the most centrally located data point in the data set. After finding a set of *k* medoids, *k* clusters are constructed by assigning each observation to the nearest medoid. Representation by *k*-medoids has the advantages that it presents no limitations on attribute types and it is not very sensitive to the presence of outliers, as the choice of medoids is dictated by the location of a predominant fraction of points inside a cluster (56). PAM works well for small data sets, but does not scale well with large data sets.

CLARA (*clustering large applications*) (6) is an implementation of PAM in which PAM works on different subsets of the data set. First, multiple samples of the data set are drawn, then PAM is applied on the samples, and, finally, the best clustering output of these samples is used as the output.

CLARANS (*clustering large applications based on randomized search*) (13) combines the sampling techniques with PAM. The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of *k* medoids. The *neighbor* of the current clustering is the clustering obtained after replacing a medoid. CLARANS selects a node and searches for a local minimum among a specified number of neighbors. If a better neighbor is found, CLARANS moves to the neighbor's node and repeats the process; otherwise, the current clustering is a local optimum. When the local optimum is found, CLARANS starts with a new randomly selected node in search of a new local optimum.

5.2.3. k-Nearest-Neighbor Clustering

k-nearest neighbors (14) is a supervised clustering algorithm. It classifies new, unlabeled patterns based on training samples. The parameter *k*, which accounts for the number of patterns or (training points) closest to the query point, must be specified. The classification is performed using majority vote among the classification of the *k* patterns. It works based on the minimum distance from the query instance to the training samples to determine the *k*-nearest neighbors. The *k*-nearest neighbors, each already assigned to a class, is used to make a majority vote. The class labeled that is assigned to the highest number of these *k*-nearest neighbors is assigned to the query instance. A major drawback is that one needs to maintain the training set for each classification procedure.

5.3. Density-Based Clustering

The key idea of this type of clustering is to group neighboring objects of a data set into clusters based on density conditions measured in terms of the local distribution of nearest neighbors. Density-based algorithms typically assign clusters in dense regions of objects in the data space that are separated by regions of low density. Density-based algorithms are capable of discovering clusters of arbitrary shapes, providing a natural protection against outliers. Some examples include DBSCAN (17) and DENCLUE (18).

5.4. Grid-Based Clustering

This type of algorithm is mainly proposed for spatial data mining and it inherits the topology from the underlying attribute space. These algorithms divide the spatial area into a finite number of rectangular cells, generating several levels of cells corresponding to different levels of resolution, and then perform all operations on the quantized spatial area, which has the advantage of limiting the search combinations. STING (23) and WaveCluster (22) are some examples of this kind of algorithm.

5.5. Fuzzy Clustering

The issue of uncertainty support in the clustering task leads to the introduction of algorithms that use fuzzy logic in their procedure. Fuzzy clustering associates each pattern with every cluster using a membership function. In fuzzy clustering, each cluster is a fuzzy set of all the patterns. So they consider that a pattern can be classified as more than one cluster, as the pattern can belong to all clusters with a degree of membership. This is particularly useful when boundaries among the clusters are not well separated and are ambiguous. Moreover, the memberships may help to discover more sophisticated relationships between a given object and the disclosed clusters.

The most popular fuzzy clustering algorithm is fuzzy *c*-means (FCM) (25), which is an extension of the classical *k*-means algorithm for fuzzy applications. Fuzzy *c*-means is better than *k*-means at avoiding local minima.

5.6. Artificial Neural Networks for Clustering

Artificial neural networks (ANNs) (57) are mathematical or computational models motivated by biological neural networks. Specifically, they attempt to mimic biological neural systems' capacity to learn. ANNs consist of an interconnected group of artificial neurons (nodes) that build an architecture capable of processing the information.

Each artificial neuron receives a number of inputs either from original data or from the output of other neurons in the network. Each input comes via a connection that has a strength or weight. Each neuron has a single threshold value. The activation of the neuron is the integrated signal obtained by weighting the sum of the inputs and then subtracting the threshold.

The architecture defining how neurons are connected together models the relationship between inputs and outputs. ANNs are adaptive systems because they learn the input–output relationship through training. Two types of training are used in ANNs: supervised learning and unsupervised learning. In supervised learning, the training data contain examples of inputs together with the corresponding outputs, and the network learns to infer the relationship between the two. In unsupervised learning, however, the training algorithms adjusts the weights between the input nodes and the output nodes in the neural network by reference to a training data set that includes input variables only.

An unsupervised learning kind of ANN called *competitive neural networks* has been recognized as a powerful tool for pattern analysis, feature extraction, and cluster analysis. This kind of ANN is single-layered. Patterns are presented at the input and are associated with the output nodes. The network based on data correlations groups similar input patterns, which represent a single output neuron, which is indeed a pattern, an extracted feature, or a cluster, respectively.

ANNs are applicable to multivariate, nonlinear problems and have the advantage that there is no need to assume an underlying data distribution, which is usually done in statistical modeling.

5.6.1. Self-Organizing Maps (SOM)

One of the most popular competitive unsupervised neural network models today is the principle of a *self-organizing map* (*SOM*) (29). The SOM network has input and output nodes. For each attribute of the record, the input layer (input nodes) has a node, each one connected to every output node (output layer). The self-organizing map describes a mapping that seeks to preserve the topological properties from the higher-dimensional input space to a lower, discrete-dimensional map space (typically two-dimensional). SOM can be considered a nonlinear generalization of principal component analysis. Each connection is associated with a weight, which determines the position of the corresponding output node. The algorithm initially populates its nodes by randomly sampling the data and then, during a training process, changes the weights in a systematic fashion, adjusting the nodes in a way that captures the distribution of the data set's variability. At the end of the training process, each output node represents the average pattern of the data that map into it and move to form a cluster. This reduction of the dimensionality in the data space makes a very interesting property when dealing with large data sets.

The fact that SOM is based on neural networks confers a series of advantages that makes it suitable to the clustering of large amounts of noisy data with outliers (30). However, in this approach, the training of the network – and therefore the clusters – depends on the number of nodes, and the number of clusters must be arbitrarily fixed from the beginning, making the recovery of the cluster structure a very complex and subjective job (30). SOM does not perform well with invariant profiles (30). Additionally, when a particular kind of profile is abundant, SOM will populate the majority of the clusters with this profile; the most interesting profiles will map in a few clusters and their resolution might be low (30). These problems and the lack of a tree structure to detect the relationship between the clusters have motivated the appearance of neuro-hierarchical approaches like the Self-Organizing Tree Algorithm (SOTA) (31) discussed below that combine the advantages of hierarchical clustering techniques and SOM.

5.6.2. Self-Organizing Tree Algorithm (SOTA)

SOTA (31) is a hierarchical neural network that grows into a binary tree topology. SOTA is based on SOM and growing cell structures (58). It offers a criterion to stop the growing of the tree based on the approximate distribution of the probability obtained by randomization of the original data set and therefore provides a statistical support for the cluster definition.

SOTA's run times are approximately linear with the number of items to be classified, making it suitable for large data sets. Also, because SOTA follows a top-to-bottom hierarchical approach, it forms higher clusters in the hierarchy before forming the lower clusters, with the ability to stop the algorithm at any level of hierarchy to obtain meaningful intermediate results.

SOTA was originally designed for phylogenetic reconstruction (31). It has since been applied to microarray expression data analysis (30), where it has been widely used to discover gene expression patterns in time-course microarray experiments.

5.7. Evolutionary Approaches for Clustering

The basic objective of search techniques is to find the global or approximate global optimum for combinatorial optimization problems. Combinatorial optimization problems usually have NP-hard complexity and need to search the solution space exponentially. Clustering algorithms organize a set of data points in *k* subsets by optimizing some criterion function, which is why clustering can be regarded as an optimization problem. Simple local search techniques, like hill-climbing algorithms, are used to find the partitions, but they cannot guarantee optimality, as they easily get stuck in the local optimum. More complex stochastic methods like evolutionary algorithms, genetic algorithms, simulated annealing, and Tabu search, or deterministic methods like deterministic annealing, can explore the solution space more efficiently (51).

Evolutionary approaches are inspired by natural evolution. They make use of evolutionary operators and a population of solutions (also called *individuals*) to obtain the globally optimal partition of the data. Candidate solutions are encoded as chromosomes. Selection, recombination, and mutation are the most commonly used evolutionary operators. An optimization function, called the *fitness function*, is used to evaluate the optimizing degree of the population, in which each individual has its corresponding fitness value. After an initial population of solutions is generated randomly, for example, thenselection, crossover, and mutation are iteratively applied to the population until the stop condition is satisfied (15).

A more detailed description of an evolutionary approach is described below (15):

1. A random population of solutions is chosen where each solution corresponds to a valid *k*-partition of the data. A fitness value, typically inversely proportional to the squared-error value, is associated with each solution.
2. The evolutionary operators' selection, recombination, and mutation are used on a subpopulation of solutions with the highest fitness value to generate the next population of solutions, and their fitness values are calculated.
3. Step 2 is repeated until some termination condition is satisfied.

The best-known evolutionary techniques are genetic algorithms (GAs) (59, 60), evolution strategies (ESs) (61), and evolutionary programming (EP) (62). Of these three approaches, GAs have been most frequently used in clustering.

Typically, solutions are binary strings in GAs. In GAs, solutions are propagated from the current generation to the next generation based on their fitness by a selection operator. This selection operator uses a probabilistic scheme to assign a higher probability of getting reproduced to the solutions with higher fitness, so that by favoring the best individuals in the next generation, the selection operator ensures their continuity in the population.

The recombination and mutation operators are responsible for introducing diversity in the population by performing perturbations in the individuals. From the variety of recombination operators in use, crossover is the most popular one. Crossover takes as input a pair of chromosomes (called *parents*) and outputs a new pair of chromosomes (called *children* or *offspring*) where parts of the parents' parameters have been interexchanged. *Mutation* takes as input an existing chromosome and complements a bit value at a randomly selected location, generating a new chromosome.

The GA algorithm proposed by Hall, Özyurt, and Bezdek can be regarded as a general scheme (34).

5.8. Biclustering

Biclustering (36) is a data mining technique that allows simultaneous clustering of the rows and columns of a matrix. It has acquired a lot of relevance in gene expression analysis, where the results of the application of standard clustering methods to genes are limited (63). Gene expression matrices have been extensively analyzed in two dimensions separately: the gene dimension and the condition (or sample) dimension, where the goal when analyzing gene expression data with ordinal cluster analysis is either grouping of genes according to their expression under multiple conditions or grouping of samples according to the expression of multiple genes. Biclustering seeks to find submatrices, that is, subgroups of genes and subgroups of conditions, where the genes exhibit highly correlated activities for each condition in the subgroup.

Biclustering algorithms usually define a priori the number of biclusters, and they assume that either (i) there is one bicluster in the data matrix, or (ii) the data matrix contains K biclusters, where K is the number of biclusters. In the latter scenario, the following biclusters may overlap.

Some approaches attempt to identify *one bicluster at a time*, others discover *one set of biclusters at a time,* and there are also algorithms that find all the biclusters at the same time.

From its simplest form, the problem of biclustering is NP-complete, requiring either a large computational effort or the use of some sort of heuristic approach to short-circuit the calculation. A number of different heuristic approaches have been used to address this problem:

– Iterative row and column clustering combination. The approach consists of separately applying the clustering algorithms to the rows and columns of the data matrix, combining the results afterwards using some sort of iterative procedure to combine the two cluster arrangements.

- Divide and conquer. The algorithm breaks the problem into several subproblems that are similar to the original problem but smaller in size. Then it solves the problems recursively, combining the solutions to get the solution to the original problem.
- Greedy iterative search. Makes a locally optimal choice, hoping that such a choice will lead to a globally good solution.
- Exhaustive bicluster enumeration applies a search restriction on the size of the biclusters to speed up the search.
- Distribution parameter identification. The biclusters are generated using a given statistical model. The aim is to identify the distribution parameters that best fit the available data, by minimizing a certain criterion through an iterative approach.

6. Assessment of the Output

6.1. Clustering Tendency

The majority of the clustering algorithms impose a clustering structure on the data set even if it does not possess a structure. Indeed, clustering applied to a data set with no naturally occurring clusters will impose an artificial and meaningless structure. Therefore, it is important to verify whether the data set has a structure before applying any clustering algorithm. The problem of verifying whether or not clusters actually exist in data is known as *clustering tendency determination* (7).

Cluster tendency has mainly focused on the problem of determining the optimal number of clusters present in the data. If the optimal clustering contains only one group, then a null tendency must be concluded (8).

6.2. Cluster Validity

Cluster validity is the assessment of a clustering procedure's output. The clustering process has no predefined classes; therefore, it is difficult to find an appropriate metric for measuring if the found cluster configuration is acceptable or not. A clustering structure is valid if it cannot have occurred either by chance or as an artifact of the clustering algorithm.

The objective of the clustering methods is to discover significant groups present in a data set. In general, they should search for clusters whose members have a high degree of similarity with each other and are well separated from the members of the other clusters. In cluster analysis, we face the problem that when different algorithms are applied to the same data set, they give different results. Moreover, most of the clustering algorithms are very sensitive to their input parameters and we must thus decide the optimal number of clusters that fits the data set.

There are three types of validation studies (19):

1. An *external* assessment of validity compares the recovered structure to an a priori structure. The results of a clustering algorithm are evaluated based on a prespecified structure imposed on a data set and reflects the intuition about the data set's clustering structure.
2. An *internal* examination of validity determines whether the clustering structure is intrinsically appropriate for the data. The results are evaluated in terms of quantities that involve the vectors of the data themselves.
3. A *relative* test compares two structures and measures their relative merit. Here the clustering structure is evaluated by comparing it to other clustering schemes, resulting in the same algorithm but with different parameter values.

These validation assessments are carried out using some validity indexes that provide a quantitative evaluation of the clustering results based on two criteria:

Heterogeneity of the clusters, also known as the cluster *cohesion* or *compactness*: The members of each cluster should be as close to each other as possible.

Separation or *intercluster distances*: The clusters themselves should be widely separated. There are three common approaches measuring the distance between two different clusters: distance between the closest member of the clusters, distance between the most distant members, and distance between the centers of the clusters.

Thus, a basic clustering approach may aim to search for a partition that minimizes intracluster distances and maximizes intercluster distances.

6.2.1. Validity Indices

These indices are used for measuring the "goodness" of a clustering result compared to other ones that were created by other clustering algorithms, or by the same algorithms but using different parameter values. They are based on geometric properties.

Hard clustering indices are often based on some geometric motivation to estimate how compact and well separated clusters are (e.g., Dunns′ index) (20). Others are statistically motivated, for example, by comparing the within-cluster scattering with the between-cluster separation (64).

6.2.1.1. The Dunn Index

The Dunn index is defined as

$$D_m = \min_{i=1,..,m} \left\{ \min_{j=i+1,...,m} \left(\frac{d(C_i, C_j)}{\max_{k=1,...,m} \mathrm{diam}(C_k)} \right) \right\} \quad [14]$$

where the dissimilarity function between two clusters C_i and C_j is

$$d(C_i,\ C_j) = \min_{x \in C_i,\ y \in C_j} d(x, y) \quad [15]$$

and the diameter of a cluster C is defined as

$$\mathrm{diam}(C) = \max_{x, y \in C} d(x, y) \quad [16]$$

If the data set contains compact and well-separated clusters, Dunn's index will be large, since the distance between clusters is expected to be large and the diameter of the cluster is expected to be small. The main disadvantages of the Dunn index are that the calculation of the index is time-consuming and the index is very sensitive to noise (as the maximum cluster diameter can be large in a noisy environment).

6.3. Cluster Stability

Cluster stability research is involved with the validity of clusters generated by a clustering algorithm. It answers whether generated clusters are true clusters or are due to chance. Recent work on cluster validity research has concentrated on a kind of *relative index* called *cluster stability*.

Cluster stability exploits the fact that when multiple data sources are sampled from the same distribution, the clustering algorithms should behave in the same way and produce similar structures.

Bagging (26), or *bootstrap aggregating*, can be used to assess stability and improve classification in terms of stability. In this ensemble method, a partitioning clustering procedure is applied to bootstrap learning sets and the resulting multiple partitions are combined by voting (65).

7. Representation of Clusters

A partition of the data set is the end product where the number of clusters and their structure are discovered. This partition shows the separability of the data points into the clusters. The notion of cluster representation was introduced in Duran and Odell (66) and subsequently studied by Diday and Simon (67) and Michalski et al. (68). They suggested the following representation schemes:

1. Represent a cluster of points by its centroid or by a set of distant points in the cluster.
2. Represent clusters using nodes in a classification tree.
3. Represent clusters by using conjunctive logical expressions, for example, the expression $[X1 > 3]$ $[X2 < 2]$.

Examples of cluster representations are shown in **Figs. 5.1 and 5.2.** Apart from these representation schemes, nowadays more sophisticated and informative representations of the clusters have been proposed. For example, these include relevance network and hierarchical clustering, including probability values (*p*-values) for each cluster using bootstrap resampling techniques (*see* **Fig. 5.3**).

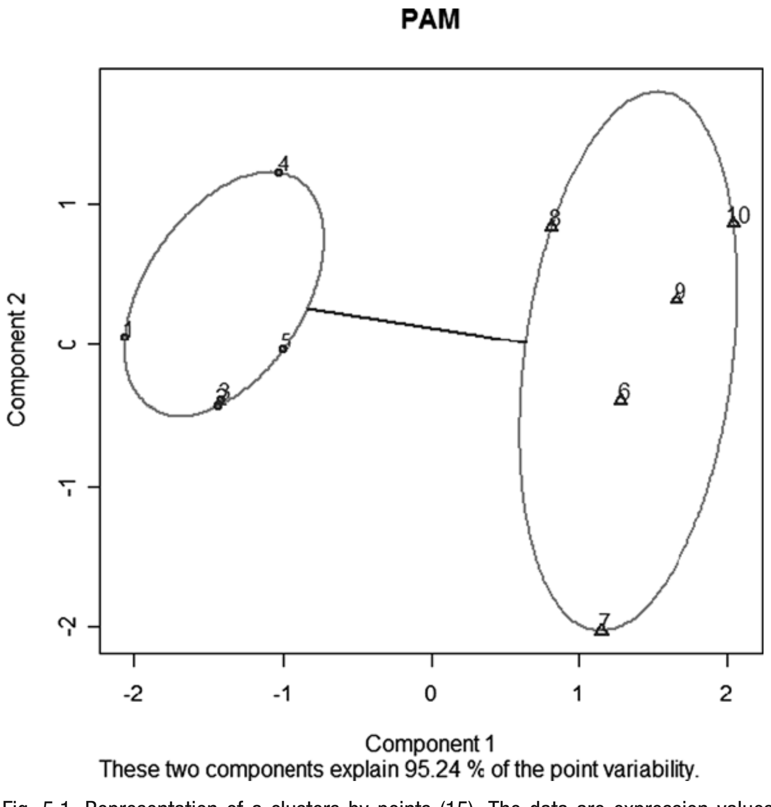
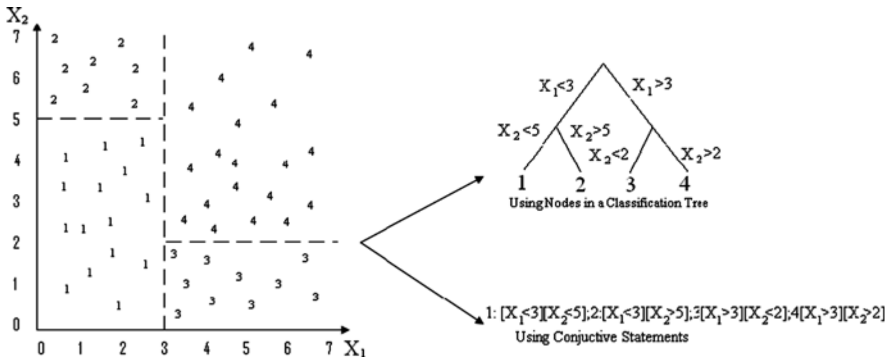


Fig. 5.1. Representation of a clusters by points (15). The data are expression values from two genes on a DNA microarray. (*Left*) From nontumor tissue; (*right*) from tumor tissues.

7.1. Relevance Networks

To explore the most relevant associations of the features, Butte and Kohane (9) proposed performing pairwise calculations of all features using a chosen similarity metric, in this case mutual information. An association with a high mutual information



X_2
X_1
$X_1<3$
$X_1>3$
$X_2<5$
$X_2>5$
$X_2<2$
$X_2>2$
1 2 3 4
Using Nodes in a Classification Tree
1: $[X_1\leq 3][X_2\leq 5]$;2:$[X_1<3][X_2>5]$;3:$[X_1>3][X_2\leq 2]$;4:$[X_1>3][X_2>2]$
Using Conjuctive Statements

Fig. 5.2. Representation of a cluster using nodes in a classification tree and representation clusters by using conjunctive logical expressions (15).

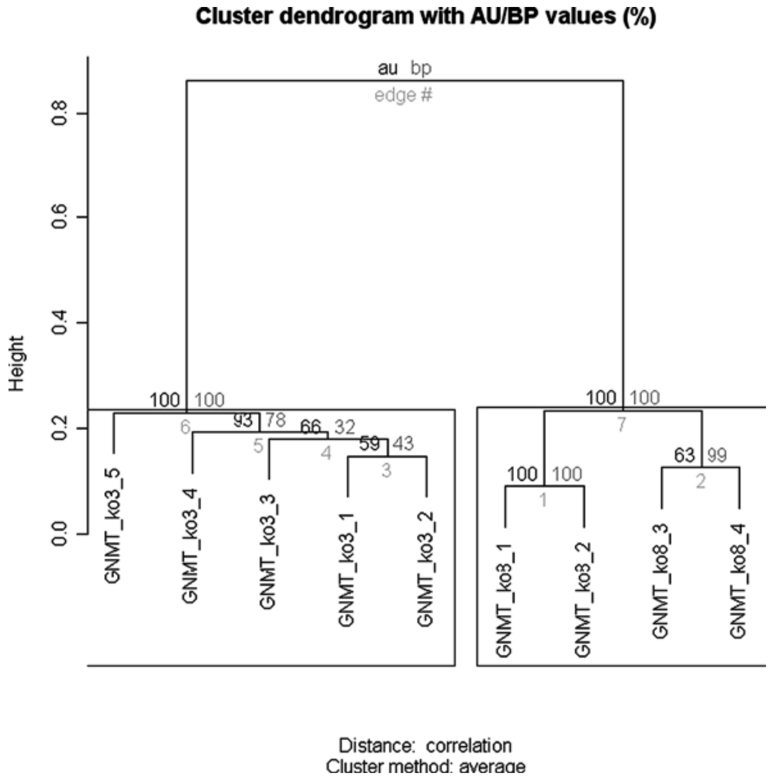


Fig. 5.3. A Pvclust output example. The example shows the hierarchical clustering of microarray data from 3- and 8-month GNMT knockout model vs. wild-type. The clustering algorithm distinguishes between the 3- and 8-month GNMT knockout models.

means that one feature is nonrandomly associated with another. They construct relevance networks by picking threshold mutual information and displaying only associations at or above the threshold.

7.2. Assessment of the Uncertainty in Hierarchical Cluster Analysis

Pvclust (27) is a package for the R statistical software that assesses the uncertainty in hierarchical cluster analysis. It calculates probability values (*p*-values) for each cluster in the dendrogram using bootstrap resampling techniques and highlights clusters with significant *p*-values. The *p*-value represents the possibility that the cluster is the true cluster. Two types of *p*-values are available: the bootstrap probability (BP) value and the approximately unbiased (AU) *p*-value. In both cases, thousands of bootstrap samples are generated by randomly sampling with replacement elements of the data, and bootstrap replicates of the dendrogram are obtained by repeatedly applying cluster analysis to them. The BP value of a cluster is the frequency it appears in the bootstrap replicates. Although the BP test is very useful in practice, it is biased (69–73). Multiscale bootstrap resampling is used for the calculation of the AU *p*-value (72, 74–76), which has superiority in bias over the BP value calculated by ordinary bootstrap resampling.

8. Notes

1. An object can be any thing, entity, or being. For example, it can be a datum, a vector, DNA, RNA, or a protein sequence.
2. Latent variables are variables inferred through a mathematical model from other variables that are observed and directly measured.
3. A metric is a nonnegative geometric function $g(x, y)$ that describes the distances between pairs of points in space.

   A metric satisfies the triangle inequality: $g(x, y) + g(y, z) \geq g(x, z)$.

   A metric should be symmetric: $g(x, y) = g(y, x)$.

   A metric also satisfies $g(x, x) = 0$.

   And lastly, it should fulfill the condition that $g(x, y) = 0$ implies $x = y$.
4. Ordinal variables do not establish the numeric difference between data points. They indicate only that one data point is ranked higher or lower than another.
5. Shannon entropy: "Information" and "uncertainty" are technical terms used to describe any process that selects one or more objects from a set of objects. If we have a device that can produce three symbols, A, B, or C, while we wait for a symbol, we are *uncertain* as to which symbol it will produce. Once a symbol appears, our uncertainty *decreases*, and we say that we have received some *information*. That is, the information is a decrease in uncertainty. The Shannon entropy is

a measure of the uncertainty, and it provides a way to estimate the minimum average message length, in bits, needed to encode a string of symbols, based on the frequency of the symbols.

6. Linkage or amalgamation rules determine how the distance between two clusters can be measured. Those include *single-linkage*, *complete-linkage*, *average-linkage,* and *minimum-variance* or *Ward's methods.* Clusters are linked sequentially to form new clusters. At each stage of this clustering process, the clusters with the shortest distance between them are combined, and the distances between the resulting set of clusters recomputed.
7. An *outlier* in statistics is a data point that does not fit a probability distribution.

References

1. Saeys Y, Inza I, Larrañaga P. (2007) *Bioinformatics* 23:2507–2517.
2. Densmore D, Heath TL. (2002) *Euclid's Elements,* Green Lion Press, Santa Fe, NM.
3. Zhang T, Ramakrishnman R, Linvy M. (1996) In *ACM SIGMOD International Conference on Management of Data*.
4. Guha S, Rastogi R, Shim K. (1998) In *ACM SIGMOD International Conference on Management of Data*.
5. Guha S, Rastogi R, Shim K. (1999) In *IEEE Conference on Data Engineering*.
6. Kaufman L, Rousseeuw P. (1990) *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, New York.
7. Gonzalez MD. (2005) In *Mathematics*, University of Puerto Rico, Puerto Rico.
8. Massey L. (2002) In *Recent Advances in Soft-Computing (RASC02)*, Nottingham, UK.
9. Butte AJ, Kohane IS. (2000) In *Pacific Symposium on Biocomputing*.
10. Krause EF. (1987) *Taxicab Geometry*, Dover Publications, Dover, UK.
11. MacQueen JB. (1967) In *5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, University of California Press, Berkeley.
12. Ball G, Hall D. (1967) *Behav Sci* 12:153–155.
13. Ng R, Han J. (1994) In *Proceedings of 20th VLDB Conference*, Santiago, Chile.
14. Lu SY, Fu KS. (1978) *IEEE Trans Syst Man Cybern* 8:381–389.
15. Jain A K. (1999) *ACM Comp Surv* 31:264–323.
16. Pearson K. (1896) *Philos Trans Roy Soc* 187:253–318.
17. Ester M, Kriegel H, Sander J, Xu X. (1996) In *2nd International Conference On Knowledge Discovery and Data Mining (KDD'96)*, pp. 226–231.
18. Hinneburg A, Keim D. (1998) In *4th International Conference On Knowledge Discovery and Data Mining (KDD'98)*, pp. 58–65.
19. Halkidi M, Batistakis Y, Vazirgiannis M. (2001) *J. Intell Inform Syst* 17: 107–145.
20. Dunn J. (1974) *J Cybern* 4:95–104.
21. Knudsen S. (2002) *A Biologist's Guide to Analysis of DNA Microarray Data*, John Wiley & Sons, New York.
22. Sheikholeslami G, Chatterjee S, Zhang A. (1998) In *Proceedings of 24th VLDB Conference*, pp. 428–439.
23. Wang W, Yang J, Muntz R. (1997) In *Proceedings of 23rd VLDB Conference.*
24. Pearson K. (1901) *Philos Mag* 2:559–572.
25. Bezdeck JC, Ehrlich R, Full W. (1984) *Comput Geosci* 10:191–203.
26. Breiman L. (1996) *Mach Learn* 24:123–140.
27. Suzuki R, Shimodaira H. (2006) *Bioinformatics* 22:1540–1542.
28. Arfken G. (1985) In *Mathematical Methods for Physicists*, Academic Press, Orlando, FL, pp. 13–18.
29. Kohonen T. (1995) *Self-Organizing Maps,* Springer-Verlag, Heidelberg, Germany.
30. Herrero J, Valencia A, Dopazo J. (2001) *Bioinformatics* 17:126–136.
31. Dopazo J, Carazo JM. (1997) *J Mol Evol* 44:226–233.

32. Spearman C. (1906) *Br J Psychol* 2:89–108.
33. Kendall M. (1938) *Biometrika* 30:81–89.
34. Hall L, Özyurt I, Bezdek J. (1999) *IEEE Trans Evol Comput* 3:103–112.
35. Shannon CE. (1948) *Bell Syst Tech J* 27:379–423 and 623–656.
36. Mirkin B. (1996) *Mathematical Classification and Clustering*, Kluwer Academic Publishers, Dordrecht, the Netherlands.
37. Bandeira LPC, Sousa JMC, Kaymak U. (2003) In *Fuzzy Sets and Systems – IFSA 2003*, Vol. 2715. Springer, Berlin.
38. Witten IH, Frank E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques*, Elsevier, San Francisco.
39. Dash M, Choi K, Scheuermann P, Liu H. (2002) In *IEEE International Conference on Data Mining (ICDM'02)*.
40. Yu L, Liu H. (2003) in *Proceedings ICML*, Washington, DC.
41. Xiong M, Fang X, Zhao J. (2001) *Genome Res* 11:1878–1887.
42. Blanco R, Larrañaga P, Inza I, Sierra B. (2004) *Int J Patt Recog. Artif Intell* 18:1373–1390.
43. Subbarao C, Subbarao NV, Chandu SN. (1995) *Environ Geol* 28:175–180.
44. Fisher RA. (1936) *Ann Eugen* 7:179–188.
45. Frank I, Friedman J. (1993) *Technometrics* 35:109–148.
46. Friedman JH, Tukey JW. (1974) *IEEE Trans Comput* 23:881–890.
47. Wold H. (1966) In *Multivariate Analysis* (Krishnaiaah PR, Ed.), Academic Press, New York, pp. 391–420.
48. Sturn A. (2000) The Institute for Genomic Research, Rockville, MD.
49. Jiang D, Tang C, Zhang A. (2004) *Trans Knowl Data Eng* 16:1370–1386.
50. Kullback S, Leibler RA. (1951) *Ann Math Stat* 22:79–86.
51. Xu R. (2005) *IEEE Trans Neural Netw* 16:645–678.
52. Johnson SC. (1967) *Psychometrika* 2:241–254.
53. Ward JH. (1963) *J Am Stat Assoc* 58:236–244.
54. Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, Dmitrousky E, Lander ES, Golub TR. (1999) *Proc Natl Acad Sci* 96:2907–2912.
55. Fung, G. (2001) A Comprehensive Overview of Basic Clustering Algorithms. Available at http://pages.cs.wisc.edu/~gfung/
56. Berkhin, P. (2002) Survey of clustering data mining techniques. Technical report, Accrue.
57. Hertz J, Krogh A, Palmer RG. (1991) *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, MA.
58. Fritzke B. (1994) *Neural Netw* 7:1441–1460.
59. Goldberg DE. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Redwood City, CA.
60. Holland JH. (1975) *Adaption in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.
61. Schewefel HP. (1981) *Numerical Optimization of Computer Models*, John Wiley and Sons, New York.
62. Fogel LJ, Owens AJ, Wals MJ. (1965) *Artificial Intelligence Through Simulated Evolution*, John Wiley and Sons, New York.
63. Madeira SC, Oliveira AL. (2004) *IEEE/ACM Trans Comput Biol Bioinform* 1:24–45.
64. Davies DL, Bouldin DW. (1979) *IEEE Trans Patt Recog Mach Intell* 1:224–227.
65. Dudoit S, Fridlyand J. (2003) *Bioinformatics* 19:1090–1099.
66. Duran BS, Odell PL. (1974) *Cluster Analysis: A Survey*, Springer-Verlag, New York.
67. Diday E, Simon JC. (1976) Clustering analysis. In *Digital Pattern Recognition*, Springer-Verlag, Secaucus, NJ.
68. Michalski R, Stepp RE, Diday E. (1981) In *Progress in Pattern Recognition* (Kanal L, Rosenfeld A, Eds.), Vol. 1, Springer-Verlag, North-Holland, New York, pp. 33–55.
69. Hillis D, Bull J. (1993) *Syst Biol* 42:182–192.
70. Felsenstein J, Kishino H. (1993) *Syst Biol* 42:193–200.
71. Zharkikh A, Li WH. (1992) *Mol Biol Evol* 9:1119–1147.
72. Efron B, Halloran E, Holmes S. (1996) *Proc Natl Acad Sci* 93:13429–13434.
73. Sanderson MJ, Wojciechwski MF. (2000) *Syst Biol* 49:671–685.
74. Shimodaira H. (2002) *Syst Biol* 51:492–508.
75. Shimodaira H. (2004) *Ann Stat* 32:2616–2641.
76. Suzuki R, Shimodaira H. (2004) In *15th International Conference on Genome Informatics.*