Cluster Analysis

Introduction

Cluster analysis, which is the most well-known example of unsupervised learning, is a very popular tool for analyzing unstructured multivariate data.

Within the data-mining community, cluster analysis is also known as **data segmentation**, and within the machine-learning community, it is also known as **class discovery**.

The methodology consists of various algorithms each of which seeks to organize a given data set into homogeneous subgroups, or clusters.

There is no guarantee that more than one such group can be found; however, in any practical application, the underlying hypothesis is that the data form a heterogeneous set that should separate into natural groups familiar to the domain experts.

Clustering is a statistical tool for those who need to arrange large quantities of multivariate data into natural groups.

- Marketers use demographics and consumer profiles in an attempt to segment the marketplace into small, homogeneous groups so that promotional campaigns may be carried out more efficiently.

- Biologists divide organisms into hierarchical orders in order to describe the notion of biological diversity; financial managers categorize corporations into different types based upon relevant financial characteristics.

- Archaeologists group artifacts (e.g., broaches) found in graves in order to understand movements of ancient peoples.
- Physicians use medical records to cluster patients for treatment diagnosis.
- Audiologists use repeated utterances of specific words by different speakers to provide a basis for speaker recognition.

Cluster analysis resembles methods for classifying items; yet the two data analytic methods are philosophically different from each other.

- First, in classification, it is known a priori how many classes or groups are present in the data and which items are members of which class or group; in cluster analysis, the number of classes is unknown and so is the membership of items into classes.

- Second, in classification, the objective is to classify new items (possibly in the form of a test set) into one of the given classes based upon experience obtained using a learning set of data; clustering falls more into the framework of exploratory data analysis, where no prior information is available regarding the class structure of the data.

- Third, classification deals almost exclusively with classifying observations, whereas clustering can be applied to clustering observations or variables or both observations and variables simultaneously, depending upon the context.

Methods for clustering items (either observations or variables) depend upon how similar (or dissimilar) the items are to each other.

Similar items are treated as a homogeneous class or group, whereas dissimilar items form additional classes or groups.

Much of the output of a cluster analysis is visual, with the results displayed as scatter plots, trees, dendrograms, silhouette plots, and heatmaps.

What Is a Cluster?

This is a difficult question to answer mainly because there is no universally accepted definition of exactly what constitutes a cluster.

As a result, the various clustering methods usually do not produce identical or even similar solutions.

A cluster is generally thought of as a group of items (objects, points) in which each item is close (in some appropriate sense) to a central item of a cluster and that members of different clusters are far away from each other.

In a sense, then, clusters can be viewed as high-density regions of some multidimensional space. Such a notion seems fine on the surface if clusters are to be thought of as convex elliptical regions.

However, it is not difficult to conceive of situations in which natural clusterings of items do not follow this pattern.

When the dimension of a space is large enough, these multidimensional items, plotted as points in that space, may congregate in clusters that curve and twist around each other; even if the various swarms of points are non-overlapping (which is unlikely), the oddly shaped configurations of points may be almost impossible to detect and identify using current techniques.

Clustering Tasks

There are numerous ways of clustering a data set of n independent measurements on each of r correlated variables.

Clustering Observations

When we speak about clustering, we usually think of clustering the n observations into groups, where the number, K, of groups is unknown and has to be determined from the data.

When analyzing microarray data, the observations may be, for example, tissue samples, disease types, or experimental conditions, and so this task is often referred to as clustering samples.

Clustering Variables

We may wish to partition the p variables into K distinct groups, where the number K is unknown and has to be determined from the data.

A group may be determined by using only one variable; however, most clusters will be formed using several variables.

These clusters should be far enough apart (in some sense) that groupings are easily identifiable. Each cluster of variables may later be replaced by a single variable representative of that cluster.

When analyzing microarray data, the variables are genes, and so we refer to this task as gene clustering.

Two-Way Clustering

Instead of clustering the variables or the observations separately, it might in certain circumstances be more appropriate to cluster them both simultaneously.

Two-way clustering is known by different names, such as block clustering or direct clustering.

This goal is especially appropriate in microarray studies, where it is desired to cluster genes and tissue samples at the same time to show which subset of genes is most closely related to which subset of disease types.

Hierarchical Clustering

There are two types of hierarchical clustering methods:

- **agglomerative**
- **divisive**

Agglomerative clustering algorithms, often called bottom-up methods,

- start with each item being its own cluster;
- then, clusters are successively merged, until only a single cluster remains.

Divisive clustering algorithms, often called top-down methods, do the opposite:

- they start with all items as members of a single cluster;
- then, that cluster is split into two separate clusters, and so on for every successive cluster, until each item is its own cluster.

Most attention in the clustering literature has been on agglomerative methods.

However, arguments have been made that divisive methods can provide more sophisticated and robust clusterings.

Dendrogram

The end result of all hierarchical clustering methods is a **dendrogram** (i.e., hierarchical tree diagram), where the k-cluster solution is obtained by merging some of the clusters from the $(k+1)$-cluster solution.

The dendrogram may be drawn horizontal or vertical, depending upon user choice or software decision; both types give the same information.

The dendrogram allows the user to read off the height of the linkage criterion at which items or clusters or both are combined together to form a new, larger cluster.

Items that are similar to each other are combined at low heights, whereas items that are more dissimilar are combined higher up the dendrogram.

Thus, it is the difference in heights that defines how close items are to each other. The greater the distance between heights at which clusters are combined, the more readily we can identify substantial structure in the data.

A partition of the data into a specified number of groups can be obtained by cutting the dendrogram at an appropriate height.

If we draw a horizontal line on the dendrogram at a given height, then the number, K, of vertical lines cut by that horizontal line identifies a K-cluster solution; the intersection of the horizontal line and one of those K vertical lines then represents a cluster, and the items located at the end of all branches below that intersection constitute the members of the cluster.

Unlike the vertical distances, which are crucial in defining a solution, the horizontal distances between items are irrelevant; the software that draws a dendrogram is generally written so that the dendrogram can be easily interpreted.

For large data sets, however, this goal becomes impossible.

Dissimilarity

The basic tool for hierarchical clustering is a measure of the **dissimilarity** or **proximity** (i.e., distance) of one item relative to another item.

Which definition of distance is used in any given application is often a matter of subjective choice.

Let $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{R}^r$. Dissimilarities usually satisfy the following three properties:

1. $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$;

2. $d(\mathbf{x}_i, \mathbf{x}_i) = 0$;

3. $d(\mathbf{x}_j, \mathbf{x}_i) = d(\mathbf{x}_i, \mathbf{x}_j)$.

Such dissimilarities are termed **metric** or **ultrametric** according to whether they satisfy a fourth property.

A metric dissimilarity satisfies

4A. $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k) + d(\mathbf{x}_k, \mathbf{x}_j)$.

An ultrametric dissimilarity satisfies

4B. $d(\mathbf{x}_i, \mathbf{x}_j) \leq \max\left\{d(\mathbf{x}_i, \mathbf{x}_k), d(\mathbf{x}_j, \mathbf{x}_k)\right\}$.

Ultrametric dissimilarities can be displayed graphically by a dendrogram.

There are several ways to define a dissimilarity, the most popular being **Euclidean distance** and **Manhattan city-block distance**.

- **Euclidean:** $d(\mathbf{x}_i, \mathbf{x}_j) = \left[(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right]^{1/2} = \left[\sum_{k=1}^{r}(x_{ik} - x_{jk})^2\right]^{1/2}$.
- **Manhattan:** $d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{r} |x_{ik} - x_{jk}|$.
- **Minkowski:** $d_m(\mathbf{x}_i, \mathbf{x}_j) = \left[\sum_{k=1}^{r} |x_{ik} - x_{jk}|^m\right]^{1/m}$.

In some applications, squared-Euclidean distance is used.

Minkowski distance includes as special cases Euclidean distance ($m = 2$) and Manhattan distance ($m = 1$).

These dissimilarity measures are all computed using raw data, not standardized data.

Standardization is usually recommended when the variability of the variables is quite different: a larger variability will have a more pronounced affect upon the clustering procedure than will a variable with relatively low variability.

A dissimilarity measure used for clustering variables is

- **1-correlation:** $d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \rho_{ij} = 1 - s_{ij}/(s_i s_j)$.

Note

$-1 \leq \rho_{ij} \leq 1$ is the correlation between the pair of variables X_i and X_j.

A relatively large positive value of ρ_{ij} suggests the variables are close to each other, whereas a large negative value of ρ_{ij} suggests the variables are far away from each other.

Thus, $1 - \rho_{ij}$ is taken as a measure of dissimilarity between the variables.

Given n observations, $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathcal{R}^r$, the starting point of any hierarchical clustering procedure is to compute the pairwise dissimilarities between observations and then arrange them into a symmetric, $(n \times n)$ proximity matrix, $\mathbf{D} = (d_{ij})$, where $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, with zeroes along the diagonal.

If we are clustering variables, the proximity matrix $\mathbf{D} = (d_{ij})$ is a symmetric, $(r \times r)$-matrix with ijth dissimilarity $d_{ij} = 1 - \rho_{ij}$.

Agglomerative Nesting (agnes)

The most popular of these clustering methods are referred to as single-linkage (or nearest-neighbor), complete-linkage (or farthest-neighbor), and a compromise between these two, average-linkage methods.

Each of these clustering methods is defined by the way in which two clusters (which may be single items) are combined or joined to form a new, larger cluster.

Single linkage uses a minimum-distance metric between clusters, complete linkage uses a greatest-distance metric, and average linkage computes the average distance between all pairs of items within the two different clusters, one item from each cluster.

There is also a weighted version of average linkage, where the weights reflect the (possibly disparate) sizes of the clusters in question.

Algorithm

1. Input: $\mathcal{L} = \{\mathbf{x}_i, i = 1, 2, \ldots, n\}$, n = number of clusters, each cluster of which contains one item.
2. Compute $\mathbf{D} = (d_{ij})$, the $(n \times n)$-matrix of dissimilarities between the n clusters, where $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, 2, \ldots, n$.
3. Find the smallest dissimilarity, say, d_{IJ}, in $\mathbf{D} = \mathbf{D}^{(1)}$. Merge clusters I and J to form a new cluster IJ.
4. Compute dissimilarities, $d_{IJ,K}$, between the new cluster IJ and all other clusters $K \neq IJ$. These dissimilarities depend upon which linkage method is used. For all clusters $K \neq I, J$, we have the following linkage options:

   Single linkage: $d_{IJ,K} = \min\{d_{I,K}, d_{J,K}\}$.

   Complete linkage: $d_{IJ,K} = \max\{d_{I,K}, d_{J,K}\}$.

   Average linkage: $d_{IJ,K} = \sum_{i \in IJ} \sum_{k \in K} d_{ik}/(N_{IJ} N_K)$,

   where N_{IJ} and N_K are the numbers of items in clusters IJ and K, respectively.

Algorithm

5. Form a new $((n-1)\times(n-1))$-matrix, $\mathbf{D}^{(2)}$, by deleting rows and columns I and J and adding a new row and column IJ with dissimilarities computed from step 4.

6. Repeat steps 3, 4, and 5 a total of $n-1$ times. At the ith step, $\mathbf{D}^{(i)}$ is a symmetric $((n-i+1)\times(n-i+1))$-matrix, $i=1,2,\ldots,n$. At the last step $(i=n)$, $\mathbf{D}^{(n)}=0$, and all items are merged together into a single cluster.

7. Output: List of which clusters are merged at each step, the value (or *height*) of the dissimilarity of each merge, and a dendrogram to summarize the clustering procedure.

No one of these algorithms is uniformly best for all clustering problems.

Whereas the dendrograms from single-linkage and complete-linkage methods are invariant under monotone transformations of the pairwise dissimilarities, this property does not hold for the average-linkage method.

Single-linkage often leads to long chains of clusters, joined by singleton points near each other, a result that does not have much appeal in practice, whereas complete-linkage tends to produce many small, compact clusters.

Average linkage is dependent upon the size of the clusters, whereas single and complete linkage, which depend only upon the smallest or largest dissimilarity, respectively, do not.

Divisive Analysis (diana)

The idea is that at each step, the items are divided into a splinter group (say, cluster A) and the remainder (say, cluster B).

The splinter group is initiated by extracting that item that has the largest average dissimilarity from all other items in the data set; that item is set up as cluster A.

Given this separation of the data into A and B, we next compute, for each item in cluster B, the following two quantities:

(1) The average dissimilarity between that item and all other items in cluster B.

(2) The average dissimilarity between that item and all items in cluster A.

Then we compute (1) minus (2) for each item in B.

If all differences are negative, we stop the algorithm.

If any of these differences are positive (indicating that the item in B is closer on average to cluster A than to the other items in cluster B), we take the item in B with the largest positive difference, move it to A, and repeat the procedure.

This algorithm provides a binary split of the data into two clusters A and B.

This same procedure can then be used to obtain binary splits of each of the clusters A and B separately.

Nonhierarchical or Partitioning Methods

Nonhierarchical clustering methods (also known as partitioning methods) simply split the data items into a predetermined number K of groups or clusters, where there is no hierarchical relationship between the K-cluster solution and the $(K+1)$-cluster solution.

That is, the K-cluster solution is not the initial step for the $(K+1)$-cluster solution.

Given K, we seek to partition the data into K clusters so that the items within each cluster are similar to each other, whereas items from different clusters are quite dissimilar.

One sledgehammer method of nonhierarchical clustering would conceivably involve as a first step the total enumeration of all possible groupings of the items.

Then, using some optimizing criterion, the grouping that is chosen as best would be that partition that optimized the criterion.

Clearly, for large data sets (e.g., microarray data used for gene clustering), such a method would rapidly become infeasible, requiring incredible amounts of computer time and storage.

As a result, all available clustering techniques are iterative and work on only a very limited amount of enumeration.

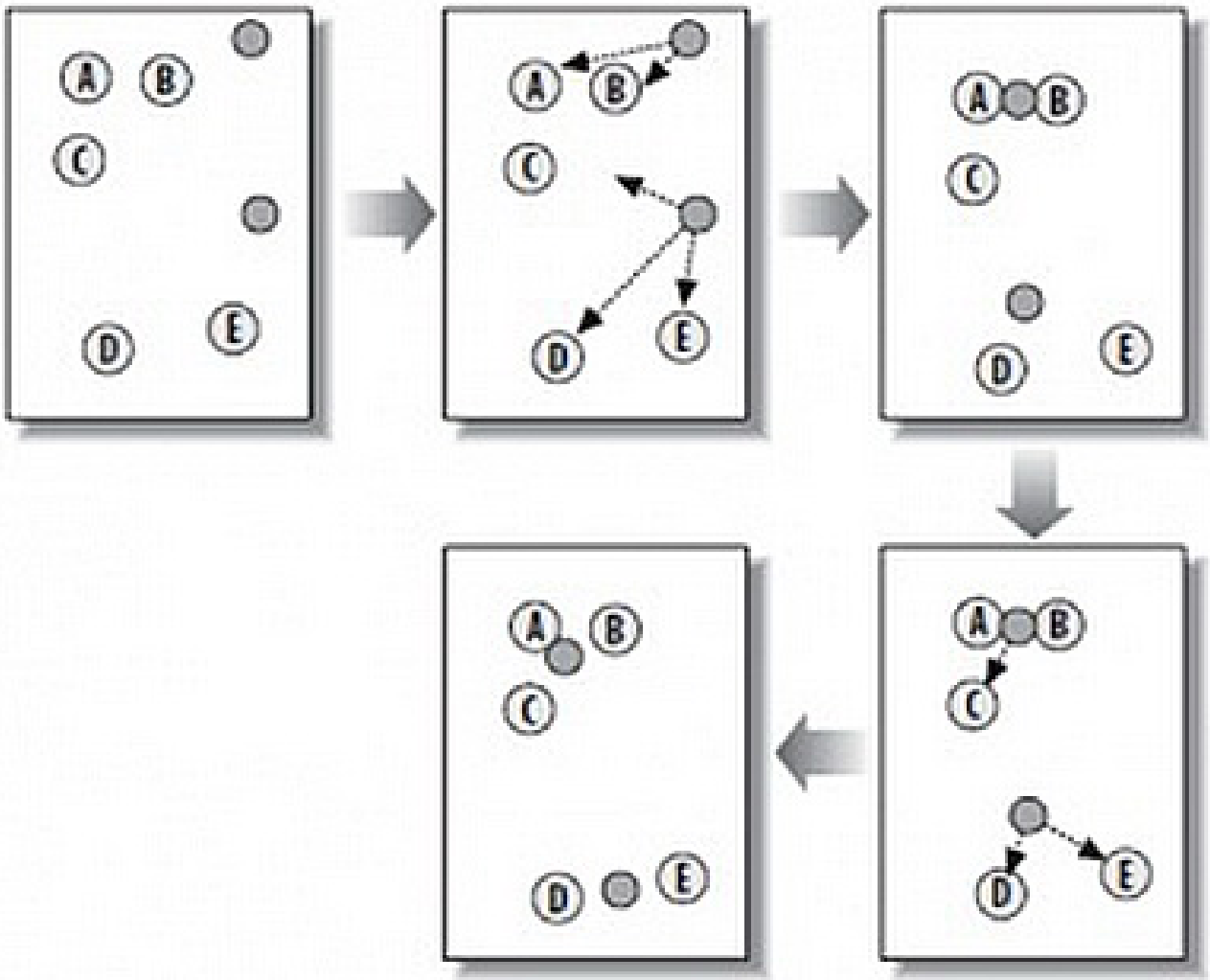
Thus, nonhierarchical clustering methods, which do not need to store large proximity matrices, are computationally more efficient than are hierarchical methods.

This category of clustering methods includes all of the partitioning methods, (e.g., K-means, partitioning around medoids) and mode-searching (or bump-hunting) methods using parametric mixtures or nonparametric density estimates.

K-Means Clustering (kmeans)

1. Input: $\mathcal{L} = \{\mathbf{x}_i, i = 1, 2, \ldots, n\}$, K = number of clusters.
2. Do one of the following:
   - Form an initial random assignment of the items into K clusters and, for cluster k, compute its current centroid, $\bar{\mathbf{x}}_k$, $k = 1, 2, \ldots, K$.
   - Pre-specify K cluster centroids, $\bar{\mathbf{x}}_k$, $k = 1, 2, \ldots, K$.
3. Compute the squared-Euclidean distance of each item to its current cluster centroid:
$$\text{ESS} = \sum_{k=1}^{K} \sum_{c(i)=k} (\mathbf{x}_i - \bar{\mathbf{x}}_k)^\tau (\mathbf{x}_i - \bar{\mathbf{x}}_k),$$
   where $\bar{\mathbf{x}}_k$ is the kth cluster centroid and $c(i)$ is the cluster containing $\mathbf{x}_i$.
4. Reassign each item to its nearest cluster centroid so that ESS is reduced in magnitude. Update the cluster centroids after each reassignment.
5. Repeat steps 3 and 4 until no further reassignment of items takes place.

K-Means Clustering (kmeans)



The K-means algorithm starts either by assigning items to one of K predetermined clusters and then computing the K cluster centroids, or by prespecifying the K cluster centroids.

The pre-specified centroids may be randomly selected items or may be obtained by cutting a dendrogram at an appropriate height.

Then, in an iterative fashion, the algorithm seeks to minimize ESS by reassigning items to clusters. The procedure stops when no further reassignment reduces the value of ESS.

The solution (a configuration of items into K clusters) will typically not be unique; the algorithm will only find a local minimum of ESS.

It is recommended that the algorithm be run using different initial random assignments of the items to K clusters (or by randomly selecting K initial centroids) in order to find the lowest minimum of ESS and, hence, the best clustering solution based upon K clusters.

Partitioning Around Medoids (PAM)

This clustering method is a modification of the K-medoids clustering algorithm.

Although similar to K-means clustering, this algorithm searches for K representative objects (or medoids) - rather than the centroids - among the items in the data set, and a dissimilarity-based distance is used instead of squared-Euclidean distance.

Because it minimizes a sum of dissimilarities instead of a sum of (squared) Euclidean distances, the method is more robust to data anomolies such as outliers and missing values.

This algorithm starts with the proximity matrix $\mathbf{D} = (d_{ij})$, where $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, either given or computed from the data set, and an initial configuration of the items into K clusters.

Using $\mathbf{D}$, we find that item (called a **representative object** or **medoid**) within each cluster that minimizes the total dissimilarity to all other items within its cluster.

In the K-medoids algorithm, the centroids of steps 2, 3, and 4 in the K-means algorithm are replaced by medoids, and the objective function ESS is replaced by ESS_{Kmed}.

Algorithm

1. Input: proximity matrix $\mathbf{D} = (d_{ij})$; K = number of clusters.

2. Form an initial assignment of the items into K clusters.

3. Locate the *medoid* for each cluster. The medoid of the kth cluster is defined as that item in the kth cluster that minimizes the total dissimilarity to all other items within that cluster, $k = 1, 2, \ldots, K$.

Algorithm

4a. For K-medoids clustering:

- For the kth cluster, reassign the i_kth item to its nearest cluster medoid so that the objective function,

$$\text{ESS}_{\text{med}} = \sum_{k=1}^{K} \sum_{c(i)=k} d_{ii_k},$$

  is reduced in magnitude, where $c(i)$ is the cluster containing the ith item.

- Repeat step 3 and the reassignment step until no further reassignment of items takes place.

4b. For partitioning-around-medoids clustering:

- For each cluster, swap the medoid with the non-medoid item that gives the largest reduction in ESS_{med}.
- Repeat the swapping process over all clusters until no further reduction in ESS_{med} takes place.

PAM

The **partitioning around medoids** (pam) modification of the K-medoids algorithm introduces a swapping strategy by which the medoid of each cluster is replaced by another item in that cluster, but only if such a swap reduces the value of the objective function.

A disadvantage of both the K-medoids and the pam algorithms is that, although they run well on small data sets, they are not efficient enough to use for clustering large data sets.

Fuzzy Analysis (fanny)

The idea behind **fuzzy clustering** is that items to be clustered can be assigned probabilities of belonging to each of the K clusters.

Let u_{ik} denote the **strength of membership** of the ith item for the kth cluster. For the ith item, we require that the $\{u_{ik}\}$ behave like probabilities; that is, $u_{ik} \geq 0$, for all i and $k = 1, 2, \cdots, K$, and $\sum_{k=1}^{K} u_{ik} = 1$ for each i.

This contrasts with the partitioning methods of K-Means or PAM, where each item is assigned to one and only one cluster.

Given a proximity matrix $\mathbf{D} = (d_{ij})$ and number of clusters K, the unknown membership strengths, $\{u_{ik}\}$, are found by minimizing the objective function,

$$\sum_{k=1}^{K} \frac{\sum_i \sum_j u_{ij}^2 u_{jk}^2 d_{ij}}{2 \sum_l u_{lk}^2}.$$

The objective function is minimized subject to the nonnegativity and unit sum restrictions by using an iterative algorithm.

Silhouette Plot

A useful feature of partitioning methods based upon the proximity matrix **D** (e.g., *K*-Means, PAM, and FANNY) is that the resulting partition of the data can be graphically displayed in the form of a **silhouette plot**.

Suppose we are given a particular clustering, $\mathcal{C}_K$, of the data into K clusters.

Let $c(i)$ denote the cluster containing the ith item.

Let a_i be the average dissimilarity of that ith item to all other members of the same cluster $c(i)$.

Also, let c be some cluster other than $c(i)$, and let $d(i, c)$ be the average dissimilarity of the ith item to all members of c.

Compute $d(i, c)$ for all clusters c other than $c(i)$.

Let $b_i = \min_{c \neq c(i)} d(i, c)$.

If $b_i = d(i, C)$, then, cluster C is called the **neighbor** of data point i and is regarded as the second-best cluster for the ith item.

The ith **silhouette value** (or **width**) is given by

$$s_i(\mathcal{C}_K) = s_{iK} = \frac{b_i - a_i}{\max\{a_i, b_i\}}, \quad -1 \leq s_{iK} \leq 1.$$

- Large positive values of s_{iK} (i.e., $a_i \approx 0$) indicate that the ith item is well-clustered.
- Large negative values of s_{iK} (i.e., $b_i \approx 0$) indicate poor clustering.
- $s_{iK} \approx 0$ (i.e., $a_i \approx b_i$) indicates that the ith item lies between two clusters.

If $\max_i\{s_{iK}\} < 0.25$, this indicates either that there are no definable clusters in the data or that, even if there are, the clustering procedure has not found it.

Negative silhouette widths tend to attract attention: the items corresponding to these negative values are considered to be borderline allocations; they are neither well-clustered nor are they assigned by the clustering process to an alternative cluster.

A **silhouette plot** is a bar plot of all the $\{s_{iK}\}$ after they are ranked in decreasing order, where the length of the ith bar is s_{iK}.

The **average silhouette width**, $\bar{s}_K$, is the average of all the $\{s_{iK}\}$.

The statistic $\bar{s}_K$ has been found to be a very useful indicator of the merit of the clustering $\mathcal{C}_K$. The average silhouette width has also been used to choose the value of K by finding K to maximize $\bar{s}_K$.

As a clustering diagnostic, Kaufman and Rousseeuw defined the **silhouette coefficient**, $SC = \max_K\{\bar{s}_K\}$, and gave subjective interpretations of its value:

SC	Interpretation
0.71–1.00	A strong structure has been found
0.51–0.70	A reasonable structure has been found
0.26–0.50	The structure is weak and could be artificial
≤ 0.25	No substantial structure has been found

Clustering Variables

We can use the same clustering methods for variables as we used for clustering observations, the main difference being the measure of distance between variables.

For clustering variables, we generally use a distance metric based upon the correlation matrix for the r variables. The correlations provide a reasonable measure of closeness between pairs of variables.

Those pairs of variables with relatively large correlations can be thought of as being close to each other; those pairs for which the corresponding correlations are small are considered to be far away from each other.

If we standardize each of the r variables to have zero mean and unit variance, then it is not difficult to show that

$$\frac{1}{2(n-1)} \sum_{i=1}^{n} (X_{ji} - X_{ki})^2 = 1 - \rho_{jk}.$$

- ρ_{jk} is the correlation between variables X_j and X_k.

This shows us that using squared Euclidean distance, $\sum_i (X_{ji} - X_{ki})^2$, is equivalent to $1 - \rho_{jk}$ as a dissimilarity measure.

Either distance metric enables us to utilize any of the hierarchical or nonhierarchical/partitioning clustering methods discussed above, and the graphical output can be a dendrogram or a silhouette plot as appropriate.

Gene Clustering

The most popular use of variable clustering has been in clustering the thousands or tens of thousands of genes measured using a microarray experiment.

Concern over the enormous volume of biological information in an organism's genome has led to the idea of grouping together those genes with similar expression patterns.

This type of clustering is referred to as **gene clustering**, where, in addition to the usual hierarchical and partitioning methods, some specialized methods have been developed.

In gene clustering, the $(r \times n)$ data matrix $\mathcal{X} = (X_{ij})$ contains the gene-expression data derived from a microarray experiment, where i indexes the row (gene), j indexes the column (tissue sample), and X_{ij} is, for example, the intensity log-ratio of the abundance of the ith gene in the experimental sample relative to some reference sample; in other words, X_{ij} is a measurement of how strongly the ith gene is expressed in the jth sample.

Because X_{ij} is the log of a ratio, it follows that those ratios with values between 0 and 1 will yield negative X_{ij}, whereas those ratios greater than 1 will yield positive X_{ij}.

For typical microarray experiments, $r \gg n$, so that matrix $\mathcal{X}$ will be vertically long and skinny.

Principal-Component Gene Shaving

See the textbook.

Block Clustering

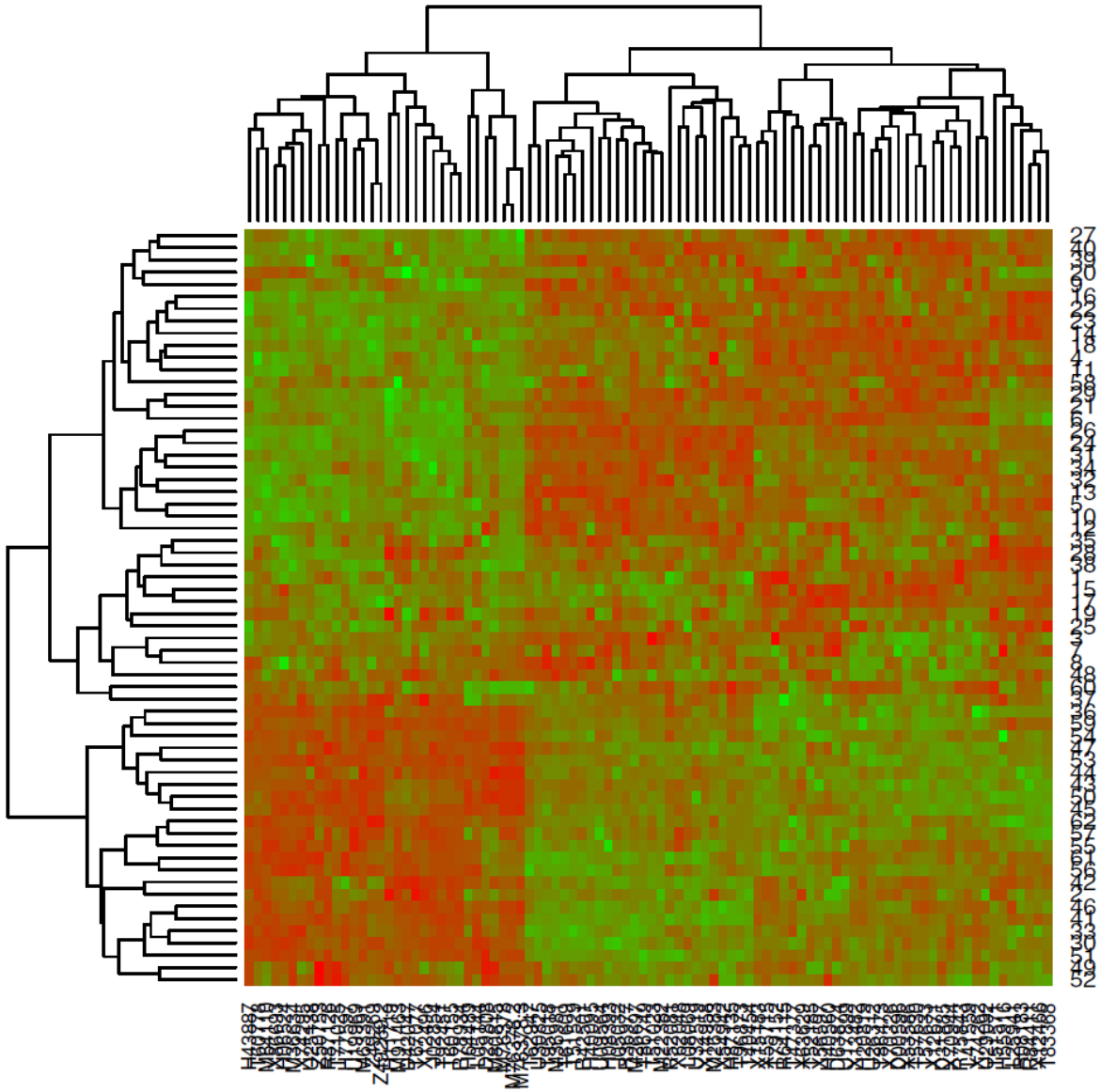
So far, our focus has been on clustering observations (cases, samples) or variables separately.

Now, we consider the problem of clustering observations and variables simultaneously.

The simplest way to do this is to apply a hierarchical clustering method to rows and columns separately. The following figure displays the heatmap of the colon cancer data, where rows and columns have been rearranged through separate hierarchical clustering algorithms.

We see a partition of the heatmap into blocks of mainly reds or greens. The rearrangement of rows (colon tissue samples) does not correspond to the known division into tumor samples and normal samples.

Heatmap



Block clustering, also known as **direct clustering**, produces a simultaneous reordering of the rows and columns of the $(r \times n)$ data matrix $\mathcal{X} = (X_{ij})$ so that the data matrix is partitioned into K submatrices or data clusters.

In block clustering, each entry in the data matrix appears in one and only one data cluster, and each data cluster corresponds to a particular row cluster and a particular column cluster.

The block-clustering algorithm given as follows partitions the rows and columns of $\mathcal{X}$ into homogeneous, disjoint blocks (i.e., where the elements of each block can be closely approximated by the same value) so that the row clusters and column clusters are hierarchically arranged to form row and column dendrograms, respectively.

Algorithm

1. Start with all data in a single block (i.e., $K = 1$).

2. Let $B_1, B_2, \ldots, B_K$ denote a partition of the rows and columns of $\mathcal{X}$ into K blocks (or data clusters), where $B_k = (\mathcal{R}_k, \mathcal{C}_k)$ consists of a set, $\mathcal{R}_k$, of r_k rows and a set, $\mathcal{C}_k$, of c_k columns of $\mathcal{X}$, $k = 1, 2, \ldots, K$.

3. Within the kth block B_k, compute $\bar{X}_k$, the average of all the X_{ij} within that block. Approximate $\mathcal{X}$ by the matrix $\widehat{\mathcal{X}} = (\widehat{X}_{ij})$, where the $\widehat{X}_{ij} = \bar{X}_k$ are constant within block B_k. Compute $\text{ESS} = \sum_{k=1}^{K} \sum_{(i,j) \in B_k} (X_{ij} - \bar{X}_k)^2$, the total within-block variance.

Algorithm

4. At the hth step, there will be h blocks, $B_1, B_2, \ldots, B_k, \ldots, B_h$. Suppose we destroy B_k by splitting it into two subblocks, B'_k and B''_k, either by splitting the rows or the columns. Consider a row-split of the block $B_k = (\mathcal{R}_k, \mathcal{C}_k)$. Suppose $\mathcal{R}_k$ *contains* a previous row-split of a different block $B_\ell = (\mathcal{R}_\ell, \mathcal{C}_\ell)$ into $B'_\ell = (\mathcal{R}'_\ell, \mathcal{C}'_\ell)$ and $B''_\ell = (\mathcal{R}''_\ell, \mathcal{C}''_\ell)$. Then, the only row-split allowable for B_k is a *fixed split* given by $\mathcal{R}'_k = \mathcal{R}'_\ell$ and $\mathcal{R}''_k = \mathcal{R}''_\ell$. Similarly for column splits. A *free split* is a split in which no such restrictions are specified.
5. The reduction in ESS due to row-splitting B_k into B'_k and B''_k is given by

$$\Delta\text{ESS} = c_k r'_k [\bar{X}(B'_k) - \bar{X}(B_k)]^2 + c_k r''_k [\bar{X}(B''_k) - \bar{X}(B_k)]^2,$$

   where $\bar{X}(B)$ denotes the average of $\mathcal{X}$ over the block B.
6. At each step, compute ΔESS for each (row or column) split of all existing blocks. Choose that split that maximizes ΔESS.
7. Stop when any further splitting leads to ΔESS becoming too small or when the number of blocks K becomes too large.