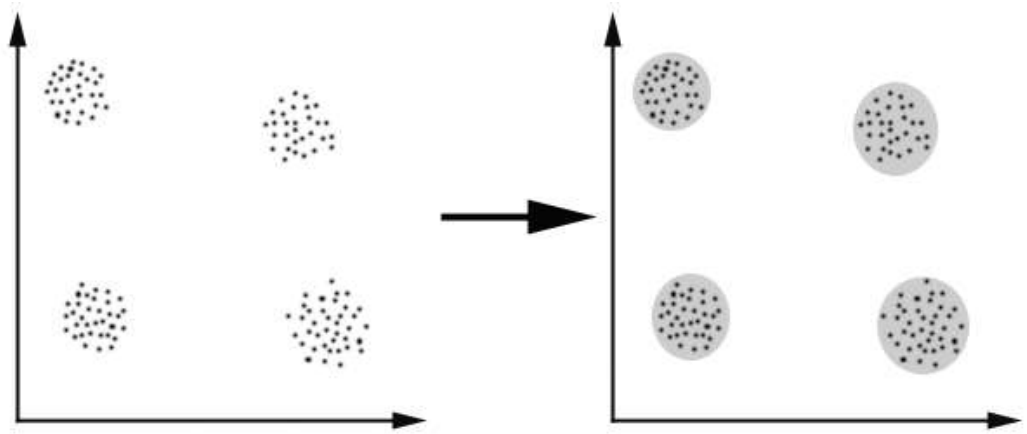
Chapter 10 : Clustering Analysis

What is Clustering?

Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data.
A loose definition of clustering could be "**the process of organizing objects into groups whose members are similar in some way**".
A *cluster* is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.



In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is *distance*: two or more objects belong to the same cluster if they are "close" according to a given distance (in this case geometrical distance). This is called ***distance-based clustering*.**

Another kind of clustering is ***conceptual clustering***: two or more objects belong to the same cluster if this one defines a concept *common* to all that objects. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

The Goals of Clustering

So, the goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs.

For instance, we could be interested in finding representatives for homogeneous groups (*data reduction*), in finding "natural clusters" and describe their unknown properties (*"natural" data types*), in finding useful and suitable groupings (*"useful" data classes*) or in finding unusual data objects (*outlier detection*).

Possible Applications

Clustering algorithms can be applied in many fields, for instance:

- ***Marketing***: finding groups of customers ~~with similar~~ behavior given a large database of customer data containing their properties and past buying records;
- ***Biology***: classification of plants and animals given their features;
- ***Libraries***: book ordering;
- ***Insurance***: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds;
- ***City-planning***: identifying groups of houses according to their house type, value and geographical location;
- ***Earthquake studies***: clustering observed earthquake epicenters to identify dangerous zones;
- ***WWW***: document classification; clustering weblog data to discover groups of similar access patterns.

Requirements

The main requirements that a clustering algorithm should satisfy are:

- scalability; حجم كبير من البيانات
- high dimensionality; زي اللي فاتت
- dealing with different types of attributes; أنواع مختلفة من البيانات
- discovering clusters with arbitrary shape; اكتشاف مجموعات بأشكال مختلفة
- minimal requirements for domain knowledge to **determine input parameters**;
- ability to deal with noise and outliers; القدرة على التعامل مع الضوضاء والحاجات الشاذة
- insensitivity to order of input records; عدم التأثر بترتيب العناصر
- interpretability and usability. التعاون والاستفادة من الهجص ده

Problems

There are a number of problems with clustering. Among them:

1. current clustering techniques **do not address all the requirements** adequately (and concurrently);
2. dealing with large number of dimensions and large number of data items can be problematic because of **time complexity**;
3. the effectiveness of the method depends on the definition of "distance" (for distance-based clustering);
4. if an *obvious* distance measure doesn't exist we must "define" it, which is not always easy, especially in multi-dimensional spaces;
5. the result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.

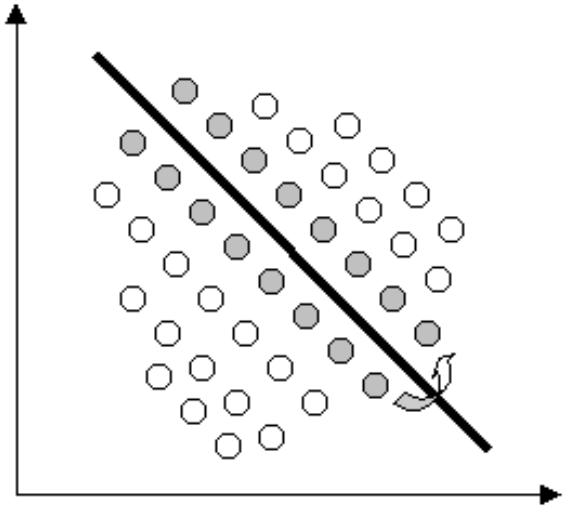
Clustering Algorithms

Clustering algorithms may be classified as listed below:

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

In the first case data are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster. A simple example of that is shown in the figure below, where the separation of points is achieved by a straight line on a bi-dimensional plane.

On the contrary the second type, the overlapping clustering, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value.



Instead, a hierarchical clustering algorithm is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted.

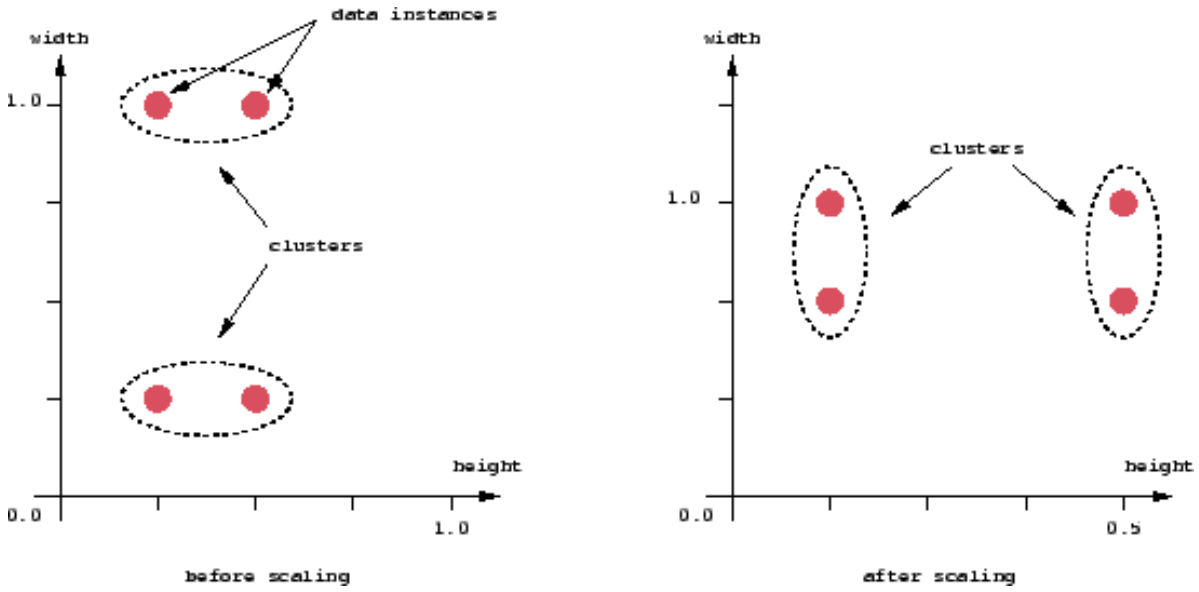
Finally, the last kind of clustering use a completely probabilistic approach.

- K-means
- Fuzzy C-means
- Hierarchical clustering
- Mixture of Gaussians

Each of these algorithms belongs to one of the clustering types listed above. So that, K-means is an *exclusive clustering* algorithm, Fuzzy C-means is an *overlapping clustering* algorithm, Hierarchical clustering is obvious and lastly Mixture of Gaussian is a *probabilistic clustering* algorithm. We will discuss about each clustering method in the following paragraphs.

Distance Measure

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances. However, even in this case the Euclidean distance can sometimes be misleading. Despite both measurements being taken in the same physical units, an informed decision has to be made as to the relative scaling. As the figure shows, different **scalings can lead to different clusterings.**



Notice however that this is not only a graphic issue: the problem arises from the mathematical formula used to combine the distances between the single components of the data feature vectors into a unique distance measure that can be used for clustering purposes: different formulas leads to different clusterings.

Again, domain knowledge must be used to guide the formulation of a suitable distance measure for each particular application.

Minkowski Metric

For higher dimensional data, a popular measure is the Minkowski metric,

$$d_p(x_i, x_j) = \left(\sum_{k=1}^{d} |x_{i,k} - x_{j,k}|^p \right)^{\frac{1}{p}}$$

where *d* is the dimensionality of the data. The ***Euclidean*** distance is a special case where *p*=2, while ***Manhattan*** metric has *p*=1. However, there are no general theoretical guidelines for selecting a measure for any given application.

It is often the case that the components of the data feature vectors are not immediately comparable. It can be that the components are not continuous variables, like length, but nominal categories, such as the days of the week. In these cases again, domain knowledge must be used to formulate an appropriate measure.

Hierarchical Clustering Algorithms

How They Work

Given a set of N items to be clustered, and an N*N distance (or similarity) matrix, the basic process of hierarchical clustering (defined by S.C. Johnson in 1967) is this:

1. Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters the same as the distances (similarities) between the items they contain.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N. (*)

Step 3 can be done in different ways, which is what distinguishes *single-linkage* from *complete-linkage* and *average-linkage* clustering.

In *single-linkage* clustering (also called the *connectedness* or *minimum* method), we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.

In *complete-linkage* clustering (also called the *diameter* or *maximum* method), we consider the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster.

In *average-linkage* clustering, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.

A variation on average-link clustering is the UCLUS method of R. D'Andrade (1978) which uses the median distance, which is much more outlier-proof than the average distance.

This kind of hierarchical clustering is called *agglomerative* because it merges clusters iteratively. There is also a *divisive* hierarchical clustering which does the reverse by starting with all objects in one cluster and subdividing them into smaller pieces. Divisive methods are not generally available, and rarely have been applied.

(*) Of course there is no point in having all the N items grouped in a single cluster but, once you have got the complete hierarchical tree, if you want k clusters you just have to cut the k-1 longest links.

Single-Linkage Clustering: The Algorithm

Let's now take a deeper look at how Johnson's algorithm works in the case of single-linkage clustering.

The algorithm is an agglomerative scheme that erases rows and columns in the proximity matrix as old clusters are merged into new ones.

The N*N proximity matrix is D = [d(i,j)]. The clusterings are assigned sequence numbers 0,1,....., (n-1) and L(k) is the level of the kth clustering. A cluster with sequence number m is denoted (m) and the proximity between clusters (r) and (s) is denoted d [(r),(s)].

The algorithm is composed of the following steps:

1. *Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.*
2. *Find the least dissimilar pair of clusters in the current clustering, say pair (r), (s), according to*

   $$d[(r),(s)] = \min d[(i),(j)]$$

   where the minimum is over all pairs of clusters in the current clustering.
3. *Increment the sequence number : m = m +1. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to*

   $$L(m) = d[(r),(s)]$$
4. *Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined in this way:*

   $$d[(k), (r,s)] = \min d[(k),(r)], d[(k),(s)]$$
5. *If all objects are in one cluster, stop. Else, go to step 2.*

An Example

Let's now see a simple example: a hierarchical clustering of distances in kilometers between some Italian cities. The method used is single-linkage.

Input distance matrix (L = 0 for all the clusters):

	BA	FI	MI	NA	RM	TO
BA	**0**	662	877	255	412	996
FI	662	**0**	295	468	268	400
MI	877	295	**0**	754	564	**138**
NA	255	468	754	**0**	219	869
RM	412	268	564	219	**0**	669
TO	996	400	138	869	669	**0**

Mito?? راجع مسلسل الحج متولي على بانوراما دراما

The nearest pair of cities is MI and TO, at distance 138. These are merged into a single cluster called **"MI/TO".** The level of the new cluster is L(MI/TO) = 138 and the new sequence number is m = 1.

distance from "MI/TO" to RM is chosen to be 564, which is the distance from MI to RM, and so on. After merging MI with TO we obtain the following matrix:

	BA	FI	MI/TO	NA	RM
BA	0	662	877	255	412
FI	662	0	295	468	268
MI/TO	877	295	0	754	564
NA	255	468	754	0	219
RM	412	268	564	219	0

min d(i,j) = d(NA,RM) = 219 => merge NA and RM into a new cluster called NA/RM

L(NA/RM) = 219

m = 2

	BA	FI	MI/TO	NA/RM
BA	0	662	877	255
FI	662	0	295	268
MI/TO	877	295	0	564
NA/RM	255	268	564	0

min d(i,j) = d(BA,NA/RM) = 255 => merge BA and NA/RM into a new cluster called BA/NA/RM, L(BA/NA/RM) = 255

m = 3

	BA/NA/RM	FI	MI/TO
BA/NA/RM	0	268	564
FI	268	0	295
MI/TO	564	295	0

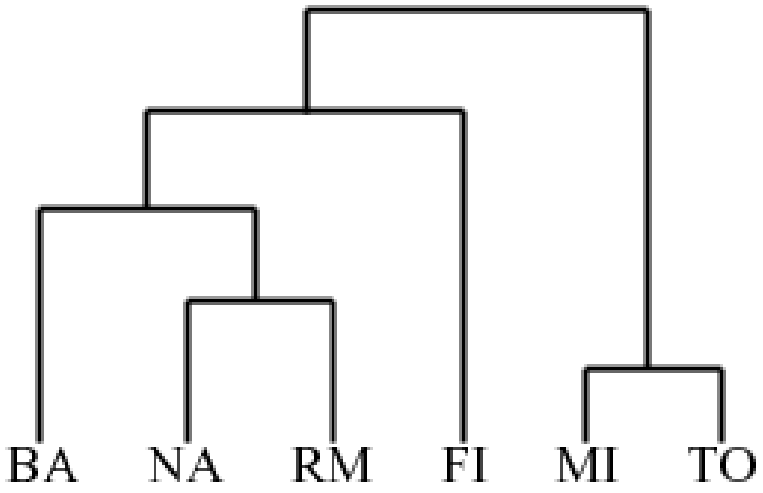
min d(i,j) = d(BA/NA/RM,FI) = 268 => merge BA/NA/RM and FI into a new cluster called BA/FI/NA/RM, L(BA/FI/NA/RM) = 268

m = 4

	BA/FI/NA/RM	MI/TO
BA/FI/NA/RM	0	295
MI/TO	295	0

Finally, we merge the last two clusters at level 295.

The process is summarized by the following hierarchical tree:



Problems

The main weaknesses of agglomerative clustering methods are:

- they do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects;
- they can never undo what was done previously.