Cluster Analysis

Cluster analysis divides data into groups (clusters) that are meaningful (Understanding), useful(Utility), or both.

Clustering for Understanding :

In the context of understanding data, clusters are potential classes and cluster analysis is the study of techniques for automatically finding classes.

For example, even relatively young children can quickly label the objects in a photograph as buildings, vehicles, people, animals, plants, etc. Other examples are..

Biology: Biologists have spent many years creating a taxonomy (hierarchical classification) of all living things.

Biologists have applied clustering to analyze the large amounts of genetic information

Clustering has been used to find groups of genes that have similar functions

Information Retrieval: The World Wide Web consists of billions of Web pages, and the results of a query to a search engine can return thousands of pages.

Clustering can be used to group these search results into a small number of clusters, each of which captures a particular aspect of the query.

For instance, a query of "movie" might return Web pages grouped into categories such as reviews, trailers, stars, and theaters.

Climate: Understanding the Earth's climate requires finding patterns in the atmosphere and ocean.

Psychology and Medicine: An illness or condition frequently has a number of variations, and cluster analysis can be used to identify these different subcategories.

For example, clustering has been used to identify different types of depression.

Business: Businesses collect large amounts of information on current and potential customers. Clustering can be used to segment customers into a small number of groups for additional analysis and marketing activities.

Clustering for Utility: **C**lustering techniques characterize each cluster in terms of a cluster prototype; i.e., a data object that is representative of the other objects in the cluster.

These cluster prototypes can be used as the basis for a number of data analysis or data processing techniques.

Summarization: Instead of applying the algorithm to the entire data set, it can be applied to a reduced data set consisting only of cluster prototypes.

Compression: Cluster prototypes can also be used for data compres- sion. **In** particular, a table is created that consists of the prototypes for each cluster.

i.e., each prototype is assigned an integer value that is its position (index) in the table.

Each object is represented by the index of the prototype associated with its cluster.

This type of compression is known as **vector quantization** and is often applied to image, sound, and video data,

where (1) many of the data objects are highly similar to one another,

(2) some loss of information is acceptable, and

(3) a substantial reduction in the data size is desired.

Efficiently Finding Nearest Neighbors:

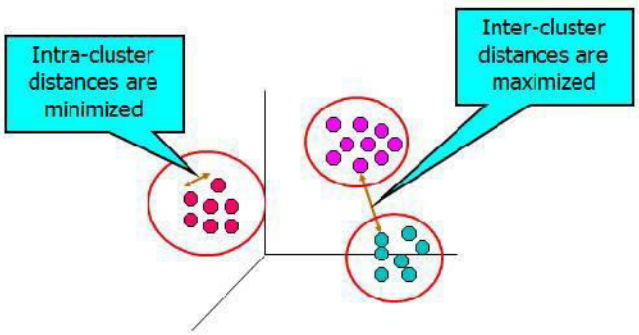
If objects are relatively close to the prototype of their cluster, then we can use the prototypes to reduce the number of distance computations that are necessary to find the nearest neighbors of an object.

Intuitively, if two cluster prototypes are far apart, then the objects in the corresponding clusters cannot be nearest neighbors of each other.

Consequently, to find an object's nearest neighbors, it is only necessary to compute the distance to objects in nearby clusters, where the nearness of two clusters is measured by the distance between their prototypes.

What is Cluster Analysis?

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



What is not Cluster Analysis?

Simple segmentation

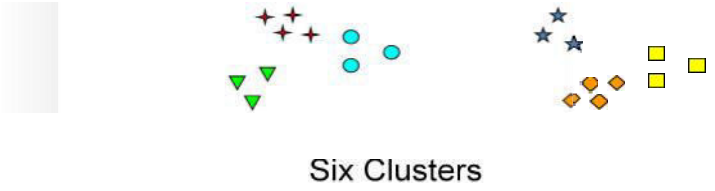
- Dividing students into different registration groups alphabetically, by last name
  - Results of a query
- Groupings are a result of an external specification
- Clustering is a grouping of objects based on the data
  - Supervised classification
- Have class label information

Notion of a Cluster can be Ambiguous:

How many clusters?

From the above figure, It is difficult to identify number of clusters because they have similar features.

It is very easy to identify number of clusters from the following examples.



Six Clusters

Two Clusters

Four Clusters

❖ Types of Clusterings:

- **Hierarchical (nested) Vs Partitional (unnested)**
- **Exclusive versus overlapping**
- **Fuzzy versus non-fuzzy**
- **Partial versus complete**
- **Heterogeneous versus homogeneous**

Partitional Clustering:

A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset

Hierarchical clustering:

A set of nested clusters organized as a hierarchical tree

Exclusive versus overlapping:

In non-exclusive(overlapping) clustering, points may belong to multiple clusters.

Can represent multiple classes or €border• points

Fuzzy versus non-fuzzy:

In fuzzy clustering, a point belongs to every cluster with some weight between 0 (absolutely doesn't belong) and 1 (absolutely belongs).

Partial versus complete:

In some cases, we only want to cluster some of the data

Heterogeneous versus homogeneous:

Clusters of widely different sizes, shapes, and densities

❖ Types of Clusters

- Well-separated clusters
- Center-based clusters (prototype based)
- Graph-Based (Contiguous clusters)
- Density-based clusters
- Property or Conceptual

Well-Separated Clusters:

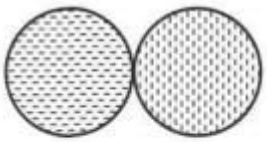
A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.

(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

Center-based (prototype based):

A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster

The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster

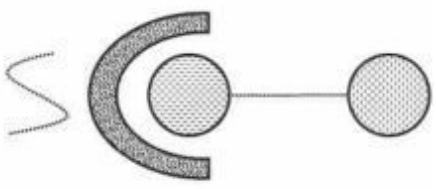


(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.

Contiguity-based Cluster (Nearest neighbor or Transitive):

A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.

Following figure shows a small bridge of points that can merge two distinct clusters.



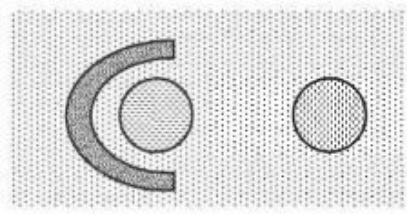
(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.

Density-based:

A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.

Used when the clusters are irregular or when noise and outliers are present.

The two circular clusters are not merged, because the bridge between them fades into the noise. Likewise, the curves that are present in fig (c) also fades into the noise and does not form a cluster.



(d) Density-based clusters. Clusters are regions of high density separated by regions of low density.

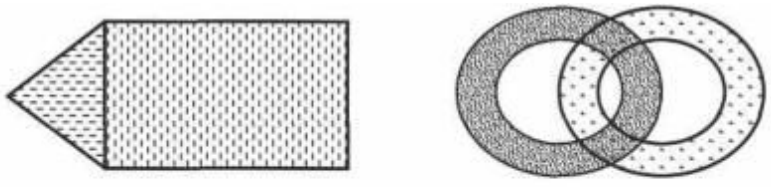
Shared Property or Conceptual Clusters

Conceptual Clusters find clusters that share some common property or represent a particular concept..

Consider the clusters shown in Figure (e). A triangular area (cluster) is adjacent to a rectangular one, and there are two intertwined circles (clusters). Points in the intersection of the circles belongs to both.

In both cases, a clustering algorithm would need a very specific concept of a cluster to successfully detect these clusters.

The process of finding such clusters is called conceptual clustering



(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)

Clustering Algorithms

- K-means and Bisecting K-means
- Hierarchical clustering
- Density-based clustering

❖ K-means Clustering:

- A Prototype based, Partitional clustering technique that attempts to find user specified number of clusters (K), which are represented by their centroids.

  Number of clusters, K, must be specified

  Each cluster is associated with a centroid (center point)

  Each point is assigned to the cluster with the closest centroid

  Algorithm:

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

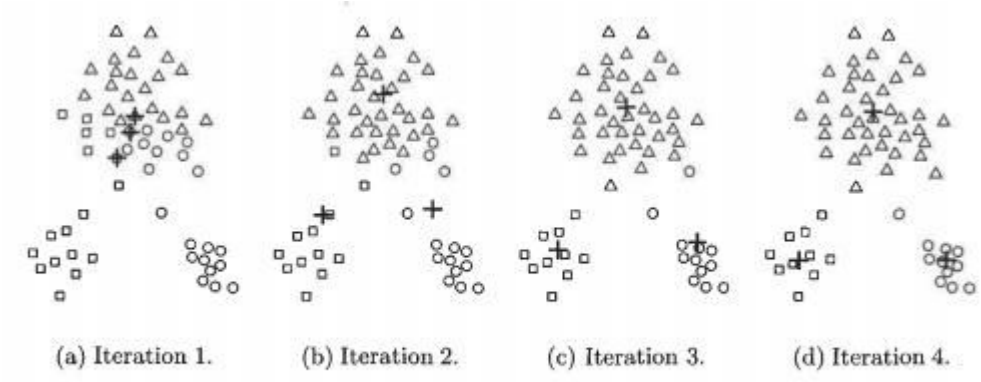


Figure 8.3. Using the K-means algorithm to find three clusters in sample data.

- Initial centroids are often chosen randomly.
- Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- €Closeness• is measured by Euclidean distance, cosine similarity, correlation, etc.

- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
- Often the stopping condition is changed to €Until relatively few points change clusters•

Complexity is O(n * K * I * d)

- n = number of points, K = number of clusters,
  I = number of iterations, d = number of attributes

Evaluation of k-means:

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two sets of clusters that are produced by two different runs of k-means, we prefer the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

❖ **K-means has problems when clusters are of differing**

- Sizes
- Densities
- Non-globular shapes
- K-means has problems when the data contains outliers.

Additional issues in k-means:

- Handling empty clusters
- Outliers
- Reducing the SSE with post processing
- Updating centroids incrementally

Handling empty clusters:

- One of the problems with the basic K-means algorithm is that empty clusters can be obtained if no points are allocated to a cluster during the assignment step.
- If this happens, then a strategy is needed to choose a replacement centroid, otherwise, the squared error will be larger than necessary.
- One approach is to choose the point that is farthest away from any current centroid.
- Another approach is to choose the replacement centroid from the cluster that has the highest SSE.
- This will typically split the cluster and reduce the overall SSE of the clustering.
- If there are several empty clusters, then this process can be repeated several times.

Outliers:

- when outliers are present, the resulting cluster centroids (prototypes) may not be as representative.
- It is often useful to discover outliers and eliminate them beforehand
- There are certain clustering applications for which outliers should not be eliminated
- When clustering is used for data compression, every point must be clustered, and in some cases, such as financial analysis, apparent outliers, e.g., unusually profitable customers, can be the most interesting points.

Reducing the SSE with post processing:

- An obvious way to reduce the SSE is to find more clusters, i.e., to use a larger *K*
- In many cases, we would like to improve the SSE, but don't want to increase the number of clusters
- Various techniques are used to "fix up" the resulting clusters in order to produce a clustering that has lower SSE.

❖ **Two strategies that decrease the total SSE by increasing the number of clusters are the following:**

- **Split a cluster:** The cluster with the largest SSE is usually chosen, but we could also split the cluster with the largest standard deviation for one particular attribute.
- **Introduce a new cluster centroid:** Often the point that is farthest from any cluster center is chosen. Another approach is to choose randomly from all points or from the points with the highest SSE.

❖ **Two strategies that decrease the number of clusters, while trying to minimize the increase in total SSE, are the following:**

- **Disperse a cluster:** This is accomplished by removing the centroid that corresponds to the cluster and reassigning the points to other clusters. Ideally, the cluster that is dispersed should be the one that increases the total SSE the least.
- **Merge two clusters:** The clusters with the closest centroids are typically chosen to merge that result in the smallest increase in total SSE.

Updating centroids incrementally:

- Instead of updating cluster centroids after all points have been assigned to a cluster, the centroids can be updated incrementally, after each assignment of a point to a cluster.
- Using an incremental update strategy guarantees that empty clusters are not produced since all clusters start with a single point, and if a cluster ever has only one point, then that point will always be reassigned to the same cluster.

Drawbacks: The clusters produced may depend on the order in which the points are processed. Incremental updates are slightly more expensive.

*****

Bisecting K-means

To obtain K clusters, split the set of all points into two clusters, select one of these clusters to split, and so on, until K clusters have been produced

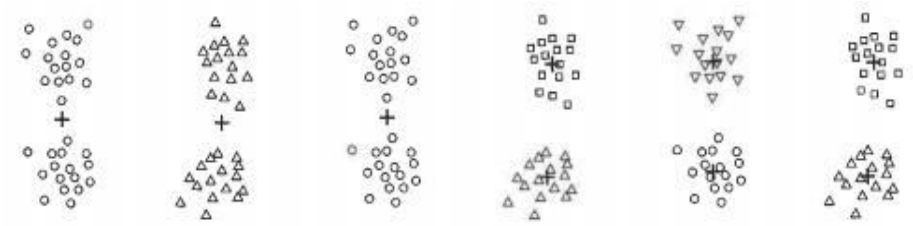
There are a number of different ways to choose which cluster to split.

We can choose the largest cluster at each step, choose the one with the largest SSE, or use a criterion based on both size and SSE. Different choices result in different clusters.

Algorithm 3 Bisecting K-means Algorithm.

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

Example:



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3.

Figure 8.8. Bisecting K-means on the four clusters example.

❖ **Strengths and weaknesses of k-means:**

- K-means is simple and can be used for a wide variety of data types.
- It is also quite efficient, even though multiple runs are often performed.
- Some variants, including bisecting K-means, are even more efficient, and are less susceptible to initialization problems.
- K-means is not suitable for all types of data.
- It cannot handle non-globular clusters or clusters of different sizes and densities.
- K-means also has trouble clustering data that contains outliers.
- Outlier detection and removal can help significantly in such situations.
- Finally, K-means is restricted to data for which there is a notion of a center (centroid). A related technique, K-medoid clustering, does not have this restriction, but is more expensive.

*****

Hierarchical Clustering

Two main types of hierarchical clustering are Agglomerative and Divisive.

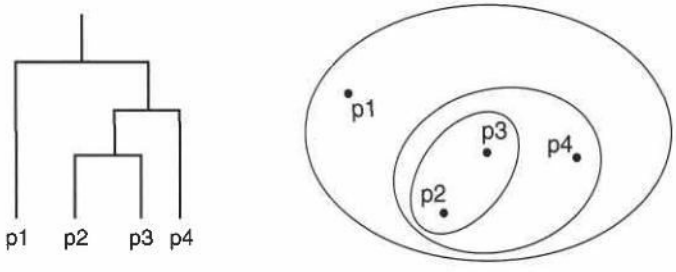
- Agglomerative:
  - Start with the points as individual clusters
  - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
- Divisive:
  - Start with one, all-inclusive cluster
  - At each step, split a cluster until each cluster contains an individual point (or there are k clusters)

Traditional hierarchical algorithms use a similarity or distance matrix

- Merge or split one cluster at a time

A hierarchical clustering is often displayed graphically using a tree-like diagram called a **dendrogram,** which displays both the cluster-subcluster relationships and the order in which the clusters were merged (agglomerative view) or split (divisive view).

For sets of two-dimensional points, a hierarchical clustering can also be graphically represented using a nested cluster diagram. Figure 8.13 shows an example of these two types of figures for a set of four two-dimensional points.



(a) Dendrogram. (b) Nested cluster diagram.

Figure 8.13. A hierarchical clustering of four points shown as a dendrogram and as nested clusters.

Agglomerative Hierarchical clustering

Basic concept: starting with individual points as clusters, successively merge the two closest clusters until only one cluster remains.

Algorithm:

Algorithm 8.3 Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until** Only one cluster remains.

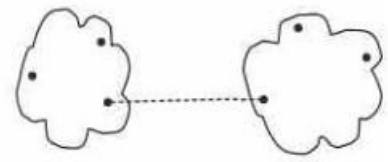
- Key operation is the computation of the proximity of two clusters
  - Different approaches are used to define the distance between clusters that distinguish the different algorithms.

Different techniques to calculate the inter cluster distance are:

- MIN
- MAX
- Group Average

❖ **MIN:**

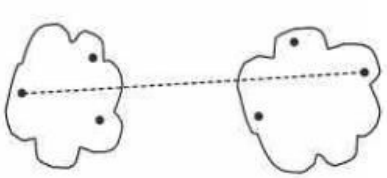
MIN defines cluster proximity as the proximity between the closest two points that are in different clusters, or using graph terms, the shortest edge between two nodes in different subsets of nodes.



(a) MIN (single link.)

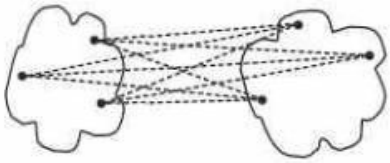
❖ **MAX:**

MAX takes the proximity between the farthest two points in different clusters to be the cluster proximity, or using graph terms, the longest edge between two nodes in different subsets of nodes.



(b) MAX (complete link.)

❖ **Group Average:**
Group average technique defines cluster proximity to be the average pairwise proximities (average length of edges) of all pairs of points from different clusters.



(c) Group average.

◆ Consider some sample data that consists of 6 two-dimensional points, which are shown in Figure 8.15. The x and y coordinates of the points and the Euclidean distances between them are shown in Tables 8.3 and 8.4, respectively

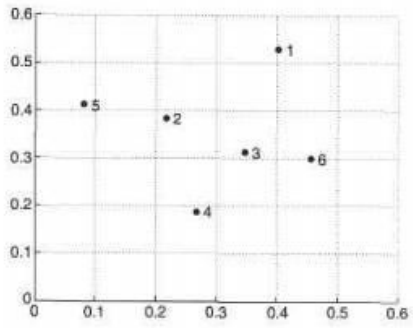


Figure 8.15. Set of 6 two-dimensional points.

Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

Table 8.3. xy coordinates of 6 points.

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

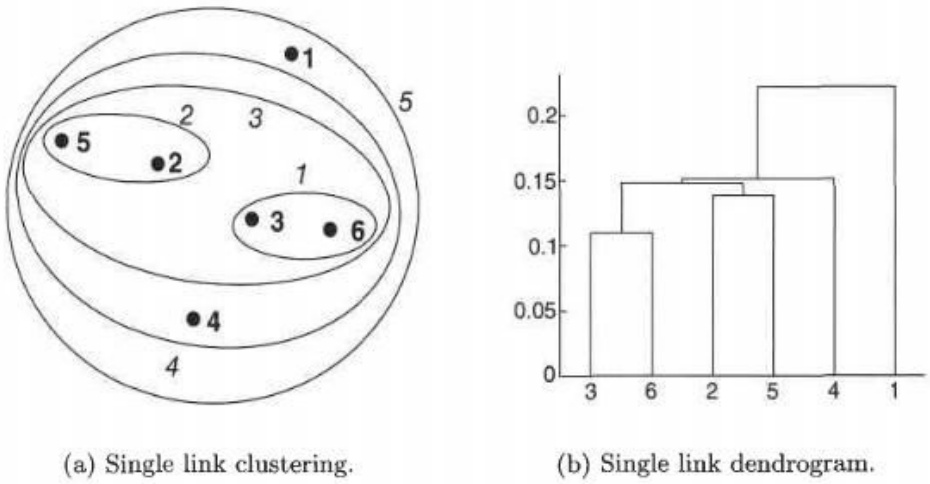
Table 8.4. Euclidean distance matrix for 6 points.

Single Link or MIN:

For the single link or MIN of hierarchical clustering, the proximity of two clusters is defined as the minimum of the distance (maximum of the similarity) between any two points in the two different clusters.

Using graph terminology, if you start with all points as singleton clusters and add links between points one at a time, shortest links first, then these single links combine the points into clusters. The single link technique is good at handling non-elliptical shapes, but is sensitive to noise and outliers.

Example: (Single Link): Figure 8.16 shows the result of applying the single link technique to our example data set of six points. Figure 8.16(a) shows the nested clusters as a sequence of nested ellipses, where the numbers associated with the ellipses indicate the order of the clustering. Figure 8.16(b) shows the same information, but as a dendrogram. The height at which two clusters are merged in the dendrogram reflects the distance of the two clusters.



(a) Single link clustering. (b) Single link dendrogram.

Figure 8.16. Single link clustering of the six points shown in Figure 8.15.

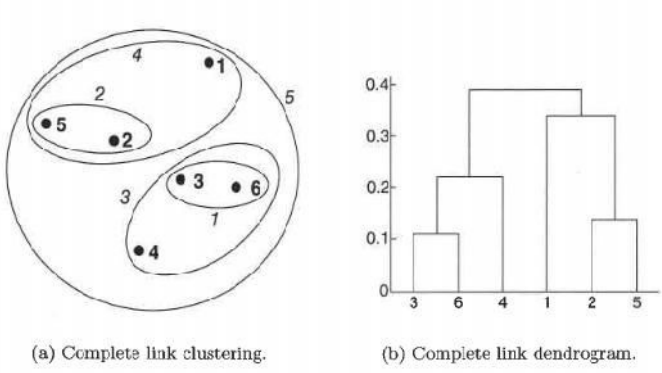
For instance, from Table 8.4, we see that the distance between points 3 and 6 is 0.11, and that is the height at which they are joined into one cluster in the dendrogram.As another example, the distance between clusters {3,6} and {2,5} is given by

$$\begin{aligned} dist(\{3,6\},\{2,5\}) &= \min(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \\ &= \min(0.15, 0.25, 0.28, 0.39) \\ &= 0.15. \end{aligned}$$

Complete Link or MAX or CLIQUE:

For the complete link or MAX version of hierarchical clustering, the proximity of two clusters is defined as the maximum of the distance (minimum of the similarity) between any two points in the two different clusters.

Using graph terminology, if you start with all points as singleton clusters and add links between points one at a time, shortest links first, then a group of points is not a cluster until all the points in it are completely linked, i.e., form a *clique*. Complete link is less susceptible to noise and outliers, but it can break large clusters and it favors globular shapes.



(a) Complete link clustering. (b) Complete link dendrogram.

Figure 8.17. Complete link clustering of the six points shown in Figure 8.15.

Example(Complete Link): Figure 8.17 shows the results of applying MAX to the sample data set of six points. {3,6} is merged with {4}, instead of {2, 5} or {1} because

$$
\begin{aligned}
dist(\{3,6\},\{4\}) &= \max(dist(3,4), dist(6,4)) \\
&= \max(0.15, 0.22) \\
&= 0.22. \\
dist(\{3,6\},\{2,5\}) &= \max(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \\
&= \max(0.15, 0.25, 0.28, 0.39) \\
&= 0.39. \\
dist(\{3,6\},\{1\}) &= \max(dist(3,1), dist(6,1)) \\
&= \max(0.22, 0.23) \\
&= 0.23.
\end{aligned}
$$

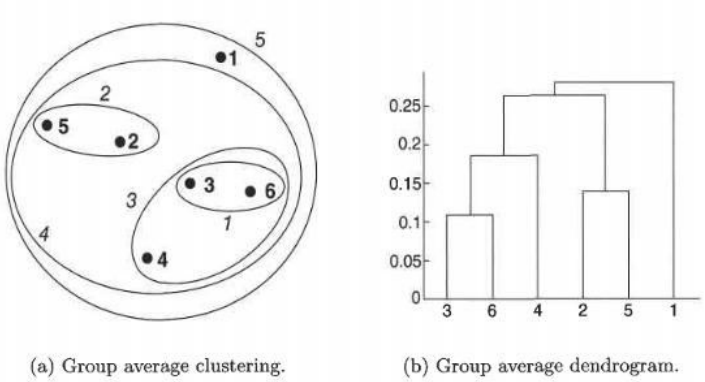
Group Average:

For the group average version of hierarchical clustering, the proximity of two clusters is defined as the average pairwise proximity among all pairs of points in the different clusters. This is an intermediate approach between the single and complete link approaches. Thus, for group average,

Proximity of two clusters is the average of pairwise proximity between points in the two clusters

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| \times |\text{Cluster}_j|}$$

Need to use average connectivity for scalability since total proximity favors large clusters



(a) Group average clustering. (b) Group average dendrogram.

Figure 8.18. Group average clustering of the six points shown in Figure 8.15.

To illustrate how group average works, we calculate the distance between some clusters

$$\begin{aligned}
dist(\{3,6,4\},\{1\}) &= (0.22+0.37+0.23)/(3*1) \\
&= 0.28 \\
dist(\{2,5\},\{1\}) &= (0.2357+0.3421)/(2*1) \\
&= 0.2889 \\
dist(\{3,6,4\},\{2,5\}) &= (0.15+0.28+0.25+0.39+0.20+0.29)/(6*2) \\
&= 0.26
\end{aligned}$$

Because $dist(\{3,6,4\},\{2,5\})$ is smaller than $dist(\{3,6,4\},\{1\})$ and $dist(\{2,5\},\{1\})$, clusters $\{3,6,4\}$ and $\{2,5\}$ are merged at the fourth stage. ■

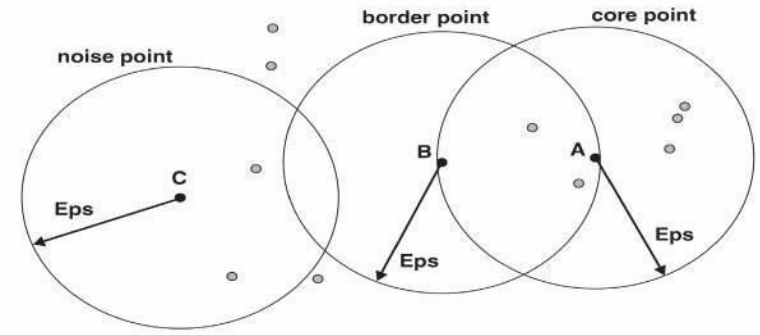
Limitations of Hierarchical clustering:

- Once a decision is made to combine two clusters, it cannot be undone
- No global objective function is directly minimized
- Different schemes have problems with one or more of the following:
    - Sensitivity to noise and outliers
    - Difficulty handling clusters of different sizes and non-globular shapes
    - Breaking large clusters

*****

DBSCAN Clustering

- Density-based clustering algorithm produces a partitional clustering.
- In this algorithm, the number of clusters is automatically determined by the algorithm.
- Points in high-density regions are considered and points in low-density regions are classified as noise and omitted. That• s why, DBSCAN does not produce a complete clustering.
- DBSCAN is a density-based algorithm.
    - Density = number of points within a specified radius (Eps)
    - A point is a core point if it has at least a specified number of points (MinPts) within Eps
        - These are points that are at the interior of a cluster
        - Counts the point itself
    - A border point is not a core point, but is in the neighborhood of a core point
    - A noise point is any point that is not a core point or a border point



The DB Scan algorithm eliminate noise points and perform clustering on the remaining points

Algorithm 8.4 DBSCAN algorithm.

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

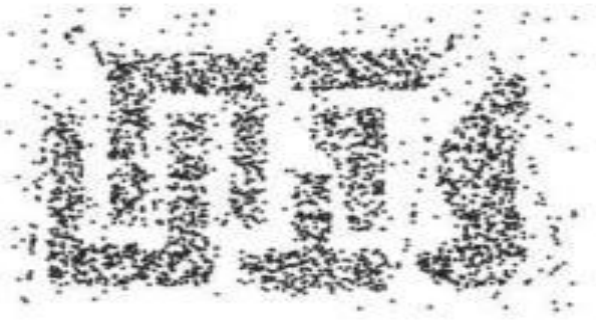
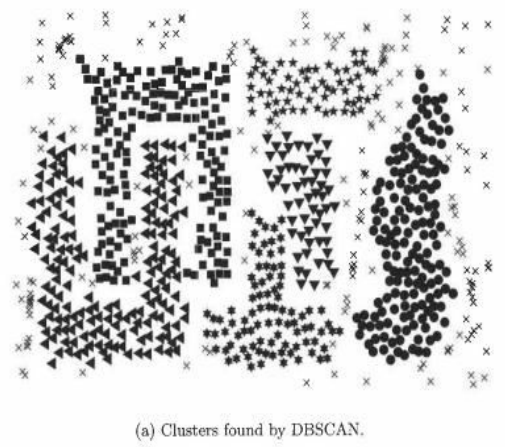
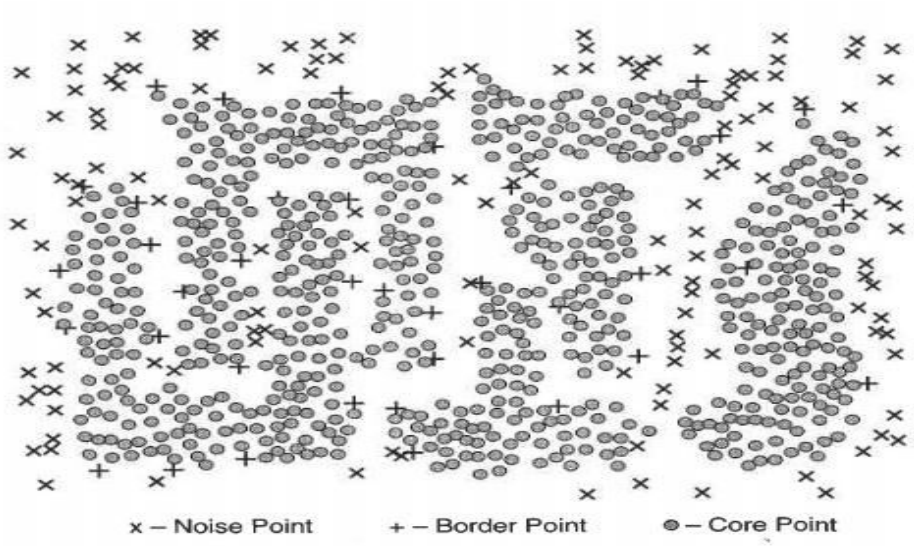


Figure 8.22. Sample data.



(a) Clusters found by DBSCAN.



(b) Core, border, and noise points.

To illustrate the use of DBSCAN, we show the clusters that it finds in the relatively complicated two-dimensional data set shown in Figure 8.22. This data set consists of 3000 two-dimensional points.

The clusters found by DBSCAN using these parameters, i.e., *MinPts* = 4 and *Eps* = 10, are shown in Figure (a). The core points, border points, and noise points are displayed in Figure (b).

Strengths and limitations

- Resistant to noise and can handle clusters of arbitrary shapes and sizes.
- DBSCAN has trouble when clusters have widely varying densities.
- It also has trouble with high dimensional data because density is more difficult to define for such data.
- DBSCAN is expensive when computation of nearest neighbors requires computing all pairwise proximities.

*****