**Unit-VI**

| Agglomerative hierarchical clustering-Basics |
| --- |
| Agglomerative hierarchical clustering-Algorithm |
| Agglomerative hierarchical clustering-Specific Techniques |
| DBSCAN –Traditional density |
| Strengths and weakness of DBSCAN |

**🡪Agglomerative hierarchical clustering-Basics:**

**Hierarchical Clustering (HC):**

-- Hierarchical Clustering is a typical clustering analysis approach via partitioning data set sequentially

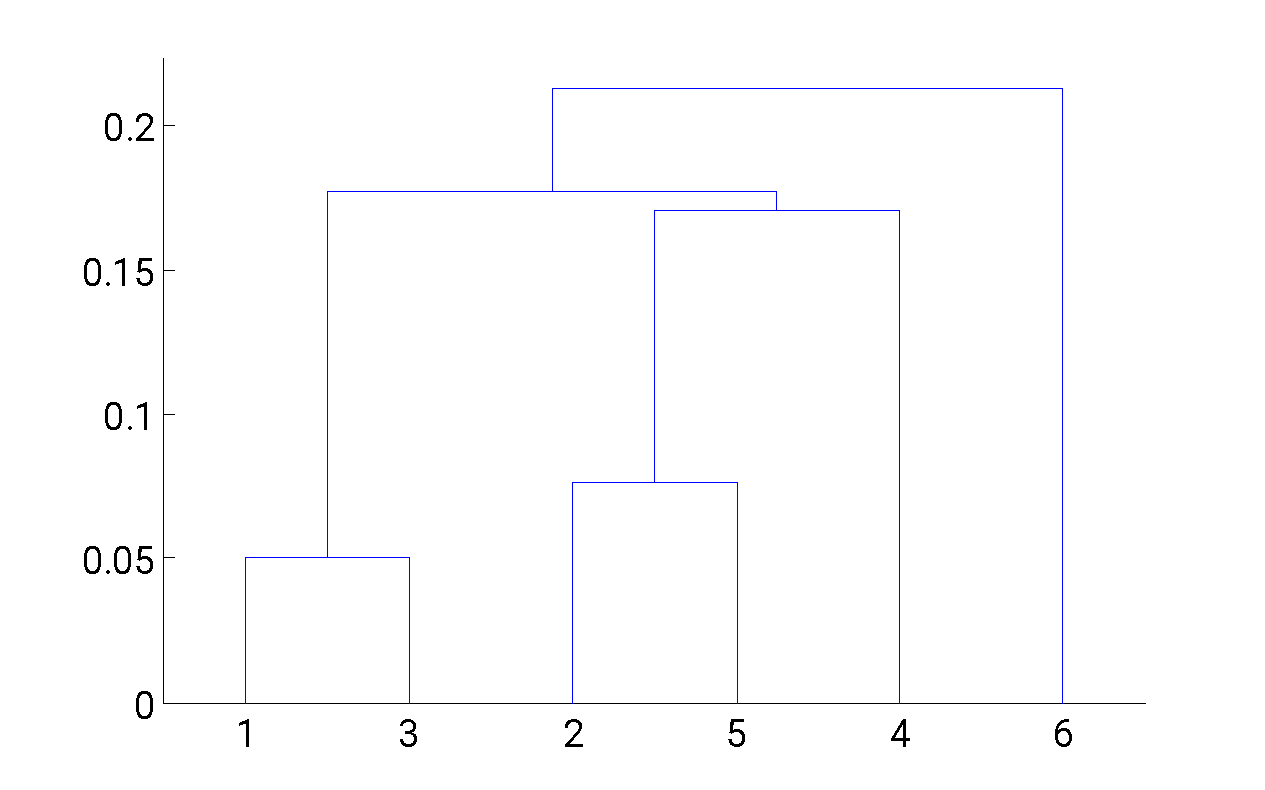
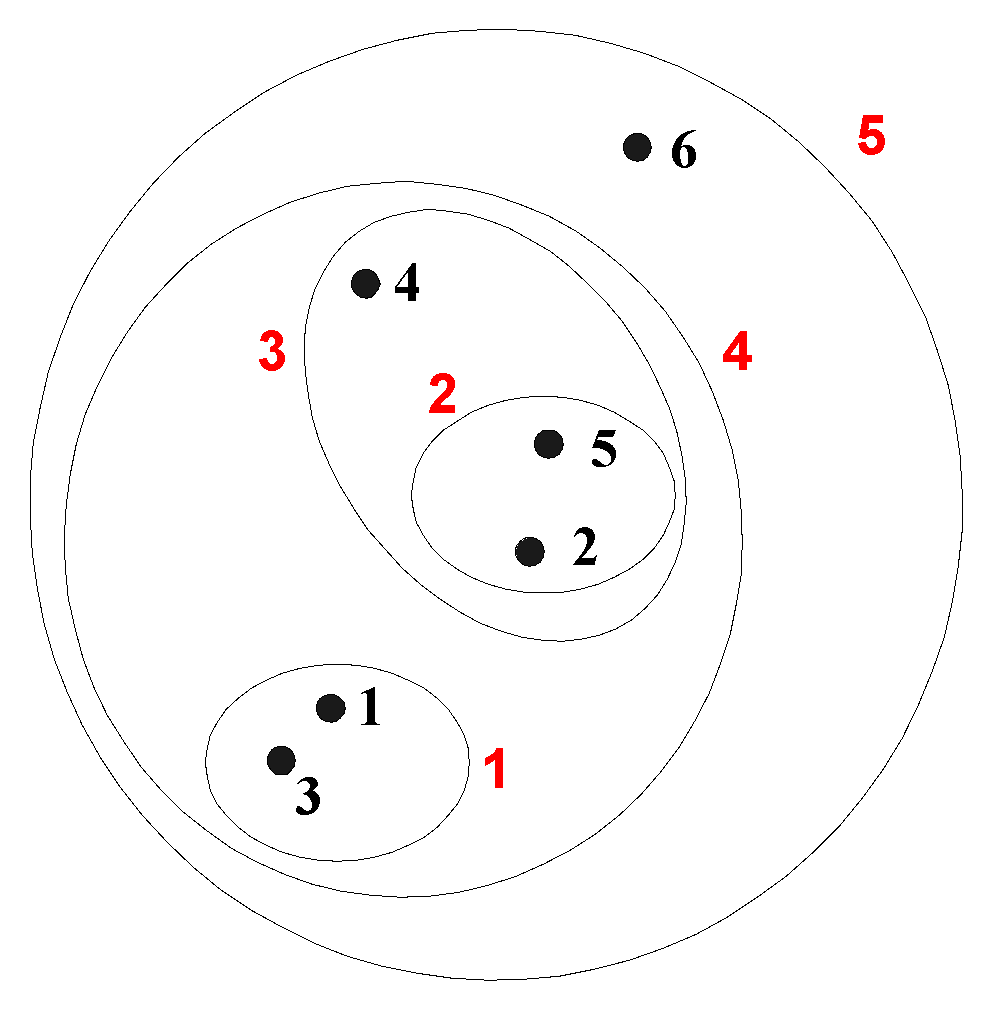
--It constructs nested partitions layer by layer via grouping objects into a tree of clusters (without the need to know the number of clusters in advance)

--This clustering technique uses distance matrix as clustering criteria

--HC (Hierarchical Clustering) produces a set of nested clusters organized as a hierarchical tree

--HC can be visualized as a dendrogram.

--Dendogram is a tree like diagram that records the sequences of merges or splits

--There are two main types of hierarchical clustering

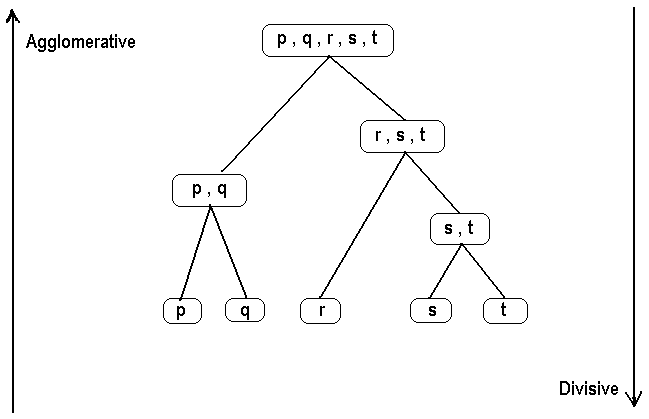
**Agglomerative Hierarchical Clustering (HAC):**

* In HAC Data objects are grouped in a bottom-up fashion.
* Initially each data object is in its own cluster.
* Then HAC merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied.
* For HAC termination condition can be specified by the user, as the desired number of clusters

**Divisive Hierarchical Clustering (DHC):**

* In this data objects are grouped in a top down manner
* Initially all objects are in one cluster
* Then the cluster is subdivided into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions as the desired number of clusters is obtained.

Agglomerative vs Divisive

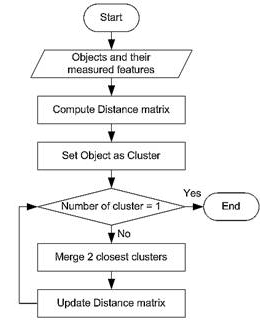


**🡪Agglomerative Clustering Algorithm:**

--More popular hierarchical clustering technique

--Basic algorithm is straightforward

* 1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains



**🡪Agglomerative hierarchical clustering-Specific Techniques:**

--MIN (single link)

Single link: smallest distance between an element in one cluster and an element in

the other, i.e., d(Ci, Cj) = min{d(xip, xjq)}

--MAX (complete link):

Complete link: largest distance between an element in one cluster and an element

in the other, i.e., d(Ci, Cj) = max{d(xip, xjq)}

--Group Average (average link):

Average: avg distance between elements in one cluster and elements in the other, i.e.,

d(Ci, Cj) = avg{d(xip, xjq)}

--Distance Between Centroids

--Other methods driven by an objective function

* + Ward’s Method uses squared error



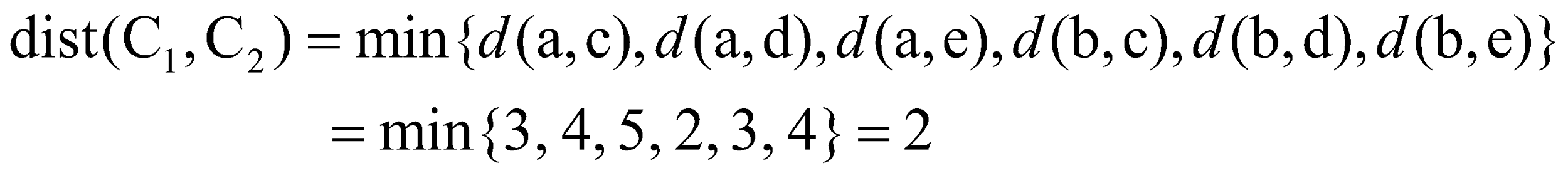
**Example**: Given a data set of five objects characterised by a single feature, assume that there are two clusters: C1: {a, b} and C2: {c, d, e}.

|  | **A** | **b** | **c** | **d** | **e** |
| --- | --- | --- | --- | --- | --- |
| Feature | 1 | 2 | 4 | 5 | 6 |

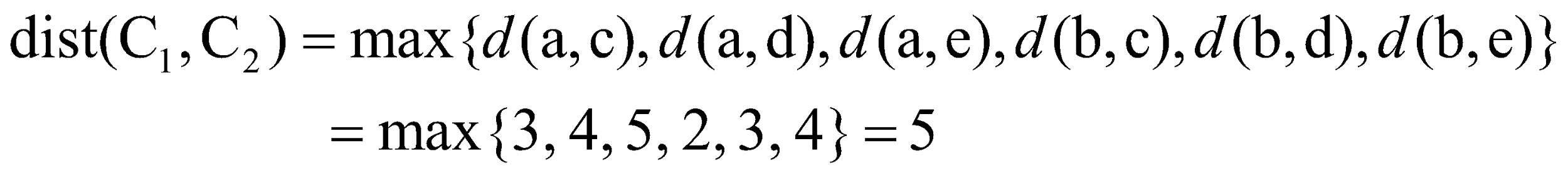
The distance matrix is:

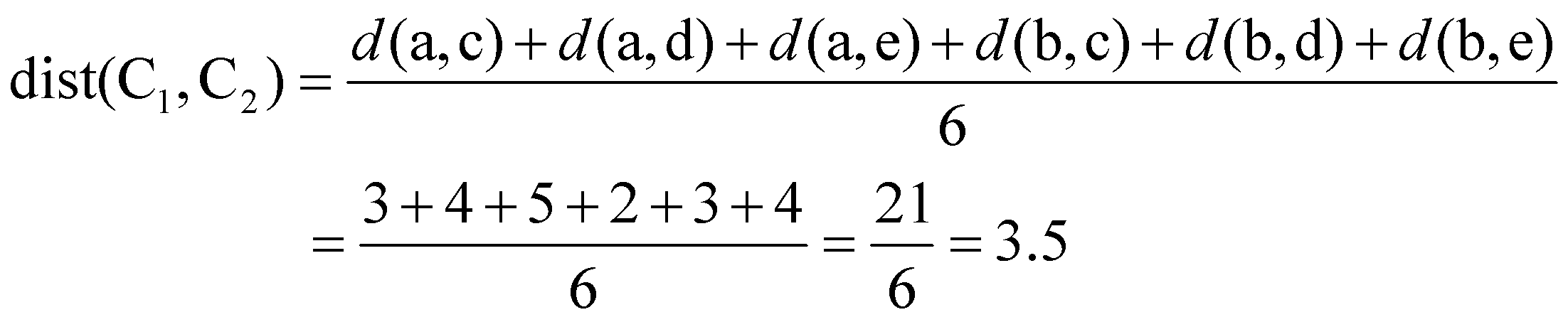
|  | **a** | **b** | **c** | **d** | **e** |
| --- | --- | --- | --- | --- | --- |
| **a** | 0 | 1 | 3 | 4 | 5 |
| **b** | 1 | 0 | 2 | 3 | 4 |
| **c** | 3 | 2 | 0 | 1 | 2 |
| **d** | 4 | 3 | 1 | 0 | 1 |
| **e** | 5 | 4 | 2 | 1 | 0 |

--The calculation of the distance between C1 and C2 is as follows:

Single link 

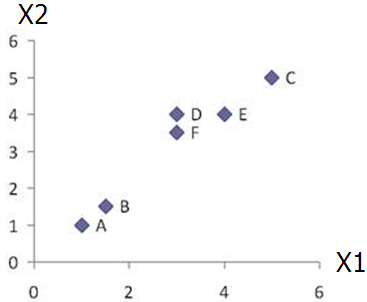
Complete link



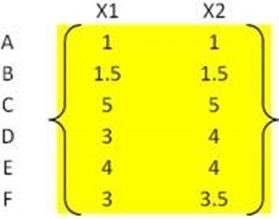
Average 

**Example: (MIN or Single Link)**

**--C**onsider the data points in the below figure:



--The Proximity matrix can be represented as:



--The distance between the points (for example A-B & D-F) can be calculated by using Euclidean distance is as follows:

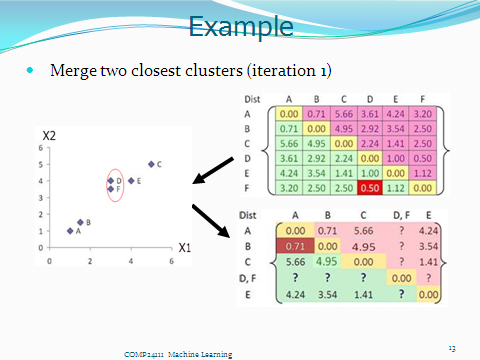




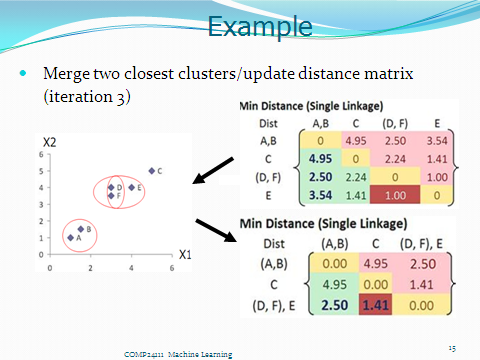
--Now the Proximity matrix representing the lowest distance is:



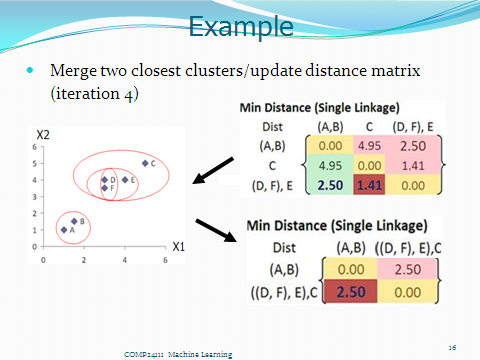
--Now we should merge two closest clusters (iteration 1)



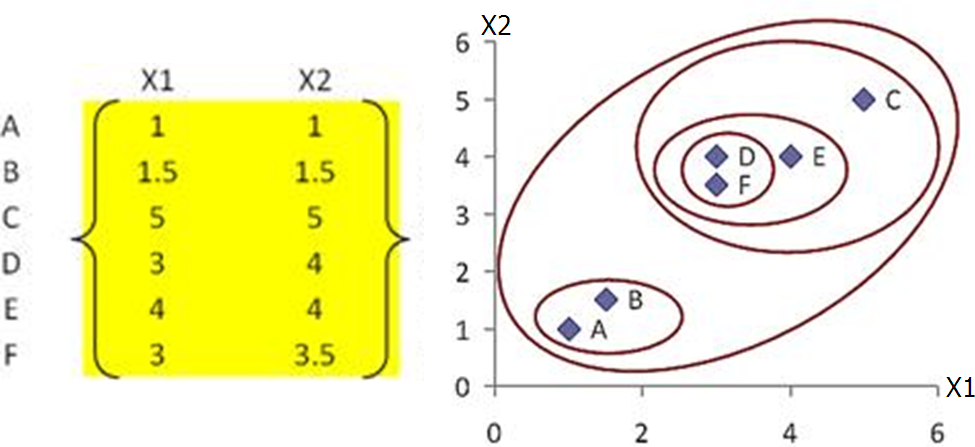
--Then we should update the proximity matrix with the newly formed cluster.



--Merge two closest clusters/update distance matrix (iteration 4)



--Final result (meeting termination condition)



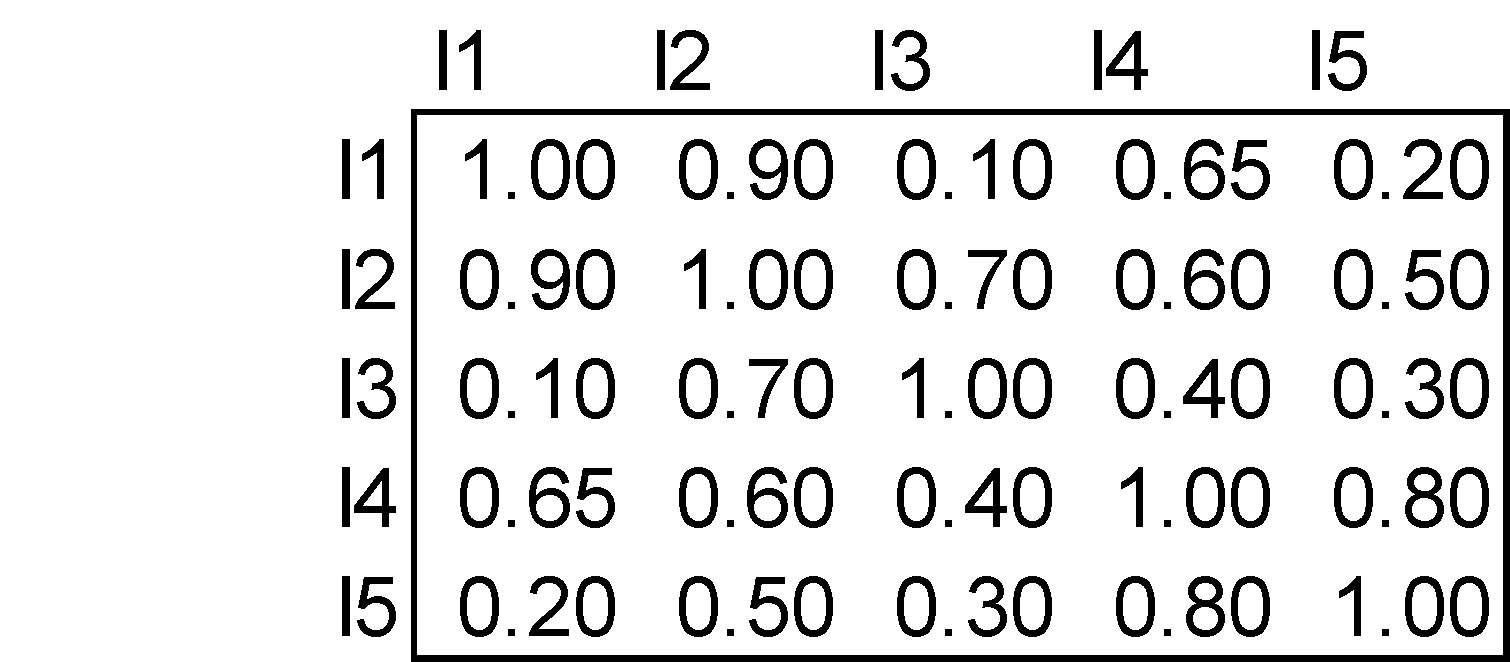
--Dendrogram tree representation

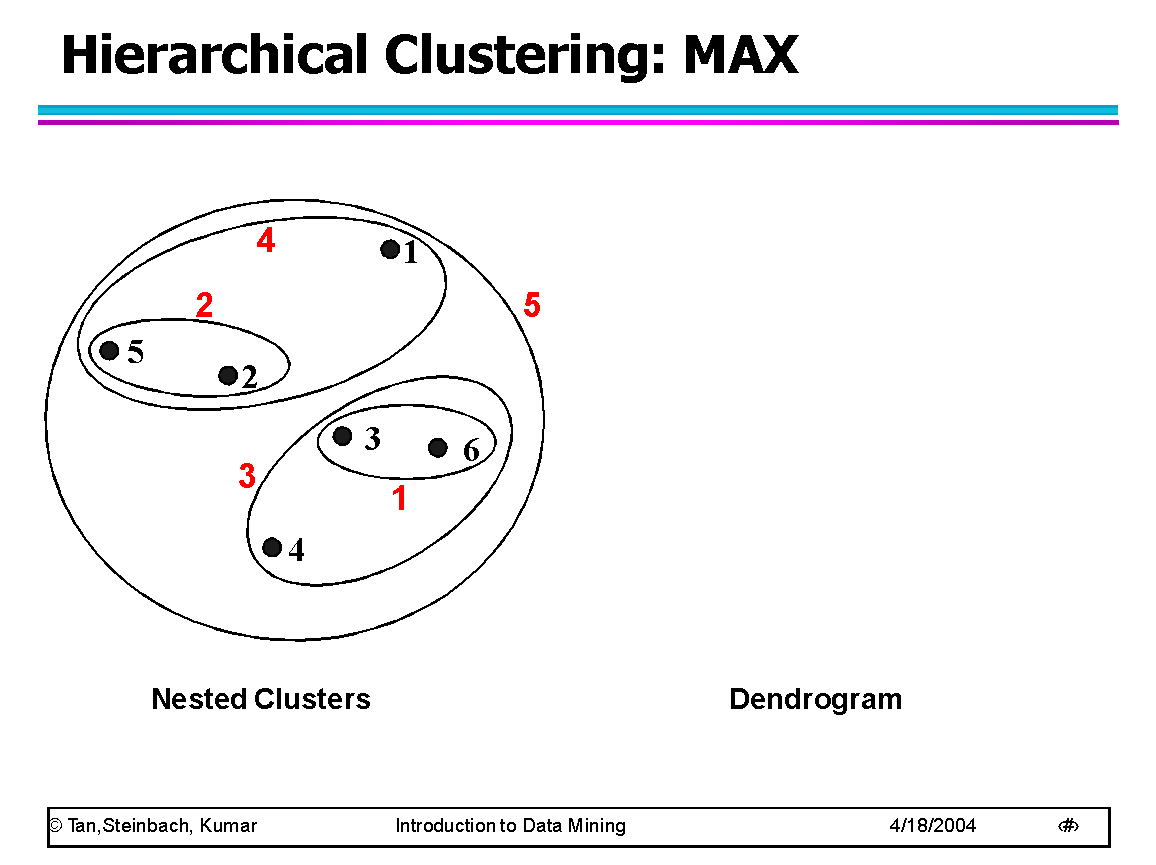


**Cluster Similarity: MAX or Complete Linkage**

--Similarity of two clusters is based on the two least similar (most distant) points in the different clusters

--Determined by all pairs of points in the two clusters





**Cluster Similarity: Group Average:**

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

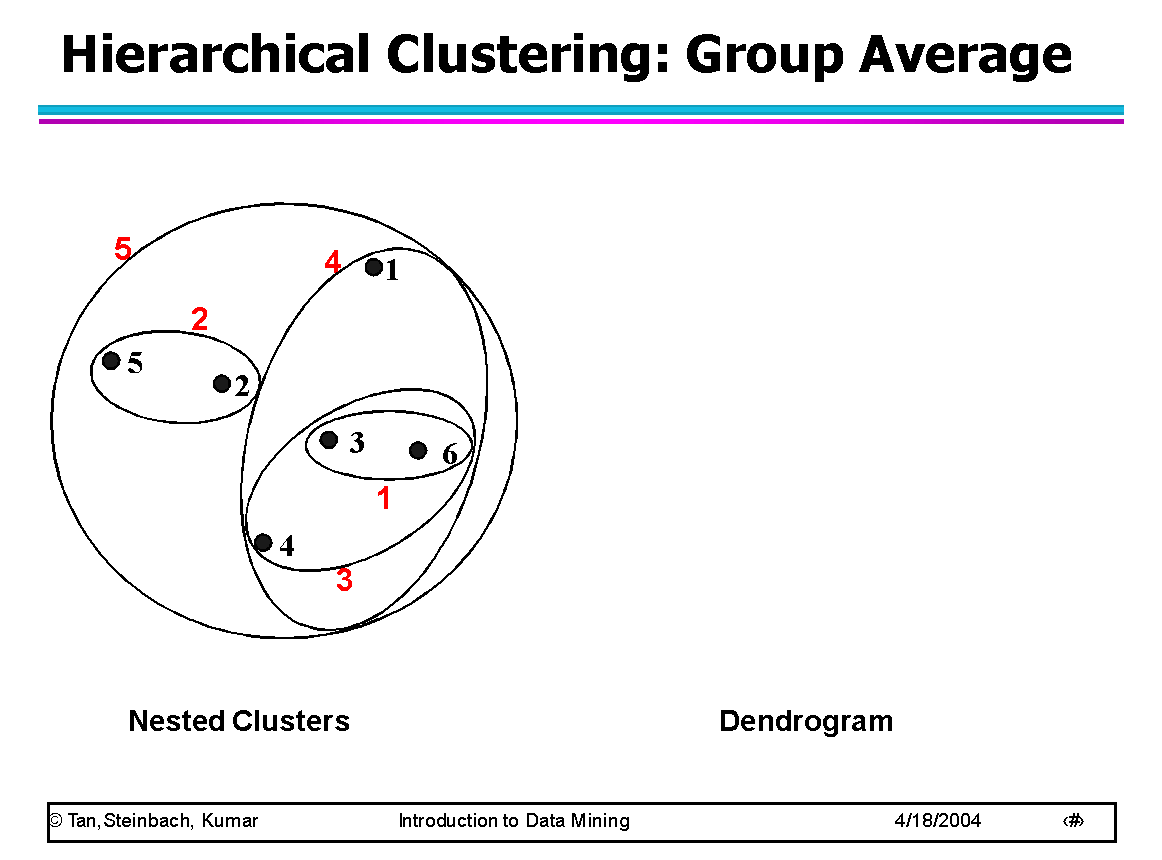
Compromise between Single and Complete Link

Strengths

* + Less susceptible to noise and outliers

Limitations

* + Biased towards globular clusters

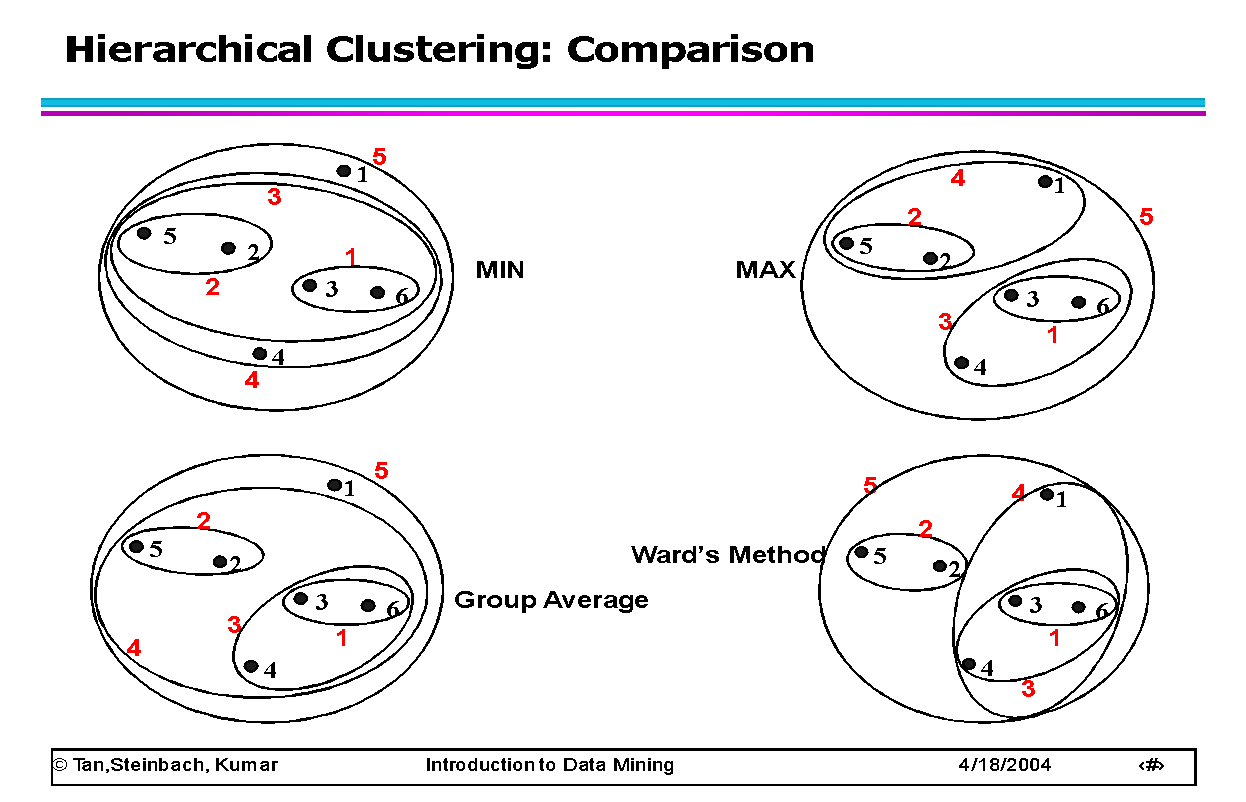


**Cluster Similarity: Wards Method:**

--Similarity of two clusters is based on the increase in squared error when two clusters are merged

--Similar to group average if distance between points is distance squared

--Less susceptible to noise and outliers



**🡪DBSCAN (Density Based SCAN):**

--DBSCAN Clustering is based on density (local cluster criterion)

--It is used to discover clusters of arbitrary shape

--DBSCAN handles noise very well.

-- DBSCAN needs only one scan to cluster the data points

--We can provide termination condition as density parameters.

--The Two parameters that we can specify are:

* + ***Eps****:* Maximum radius of neighborhood
  + ***MinPts****:* Minimum number of points in an Eps-neighborhood of a point

--The neighborhood of the data point is represented as:

***NEps(p)* =*{q Є D | dist(p,q) <= Eps}***

--The above statement can be represented as, “if the distance between ‘q’ and ‘p’ is less than or equal to the given radius, then the data point ‘q’ belongs to cluster ‘p’.

**DBSCAN Algorithm:**

Input: The data set D

Parameter: ε, MinPts

For each object p in D

if p is a core object and not processed then

C = retrieve all objects density-reachable from p

mark all objects in C as processed

report C as a cluster

else mark p as outlier

end if

End For

DBSCAN: The Algorithm

* + Arbitrary select a point *p*
  + Retrieve all points density-reachable from *p* wrt *Eps* and *MinPts*.
  + If *p* is a core point, a cluster is formed.
  + If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.
  + Continue the process until all of the points have been processed.

**Different measurements of the DBSCAN are:**

--Identifying core point, border point and outlier.

--Directly density-reachable.

--Density Connectivity.

**Identifying core point, border point and outlier:**

--A point is a core point if it has more than a specified number of points (MinPts) within Eps. These are points that are at the interior of a cluster.

--A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

--A noise point is any point that is not a core point nor a border point.

****

**Directly density-reachable:**

--A point ***p*** is directly density-reachable from a point ***q*** wrt. ***Eps***, ***MinPts*** iff

1) ***p*** belongs to ***NEps(q)***

2) ***q*** is a core point: **|*NEps (q)*|** >= ***MinPts***



**Density-Connectivity:**

--A pair of points p and q are density-connected if they are commonly density-reachable from a point o.

--Density-connectivity is symmetric



**🡪Strengths and weakness of DBSCAN:**

**Strengths:**

-- Resistant to Noise

--Can handle clusters of different shapes and sizes

**Weakness:**

--DBSCAN is sensitive to Parameters like ‘*€’ (*Radius) and ‘minpoints”.

--The size and shape of the clusters vary from one another based on the parameters given.

--DBSCAN does not work well with varying densities.

--It does not work well with High-dimensional data.

