



# Clustering

## *Knowledge Discovery and Data Mining Techniques*



# Summary

- Introduction
- Clustering basics
- Clustering Algorithms
  - Hierarchical Clustering
  - Partitional Clustering
  - Density Based Clustering
- Conclusions



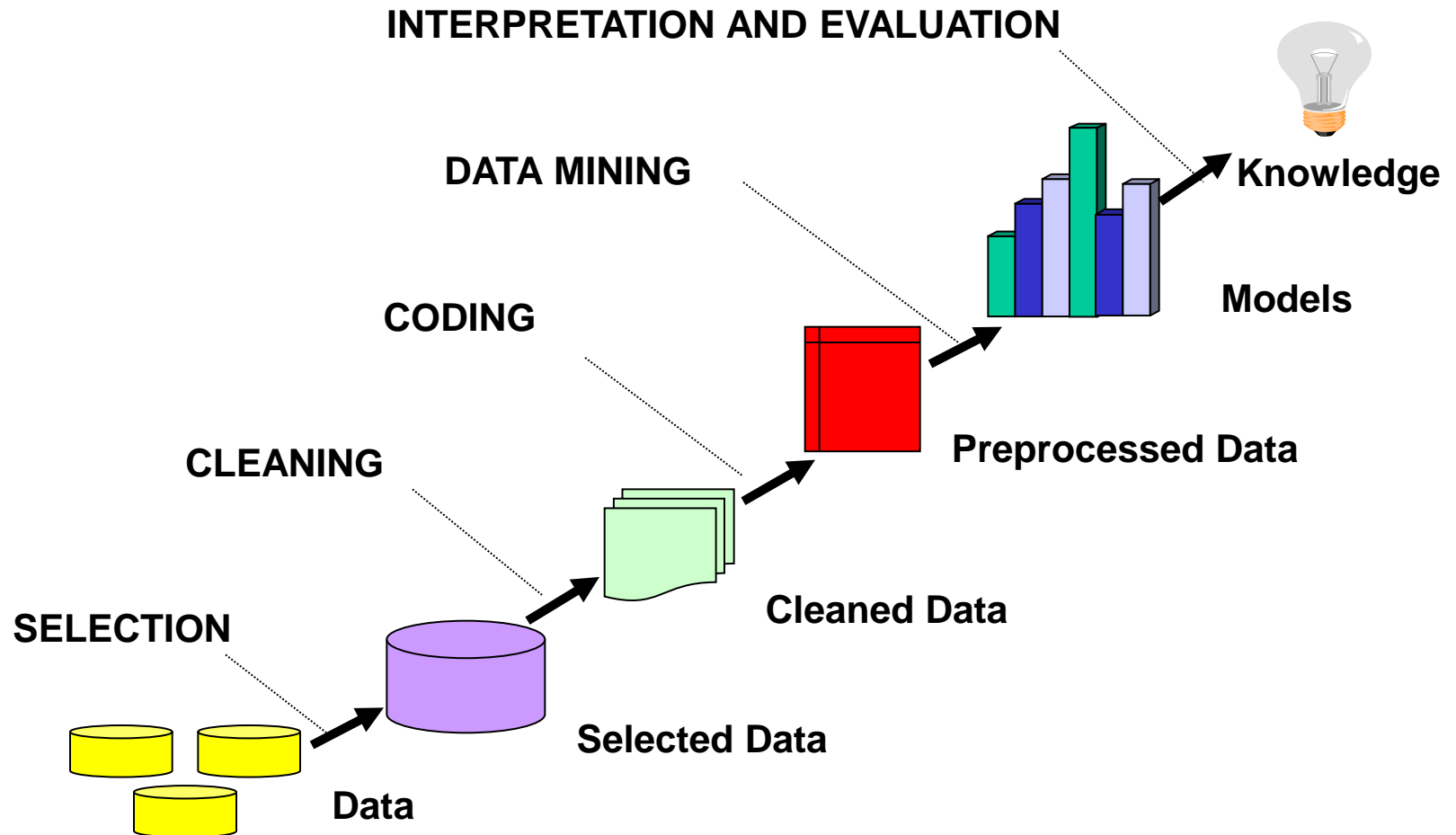
# I: Introduction



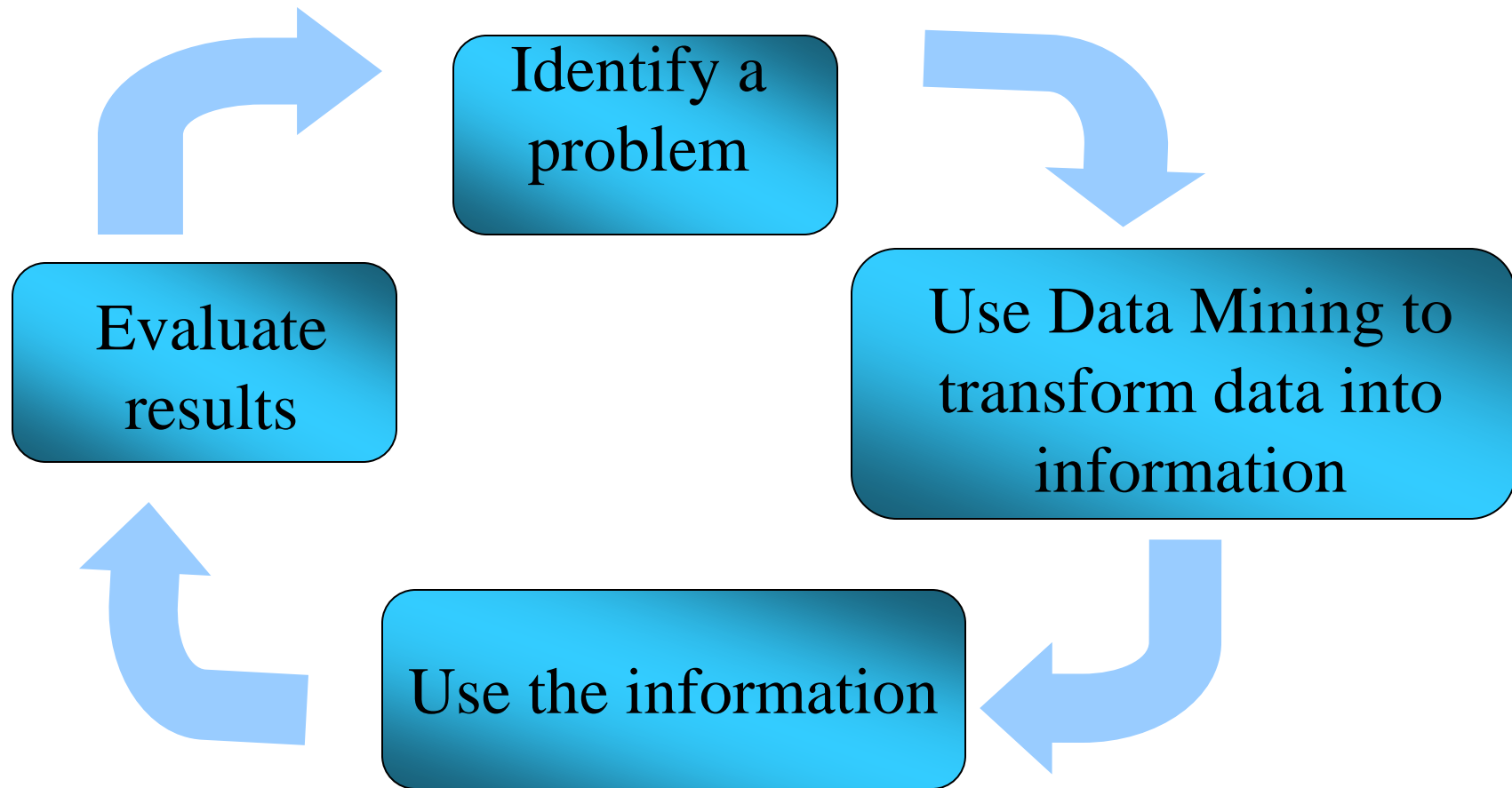
# ¿Why Clustering?

- Real world Business and organizations collect data from processes.
- Data collected can be analyzed and transformed into useful information through Data Mining
- Real world business need to learn from data to improve the relation with their clients.

# KDD Process



# Data Mining Cycle





## II: Clustering Basics



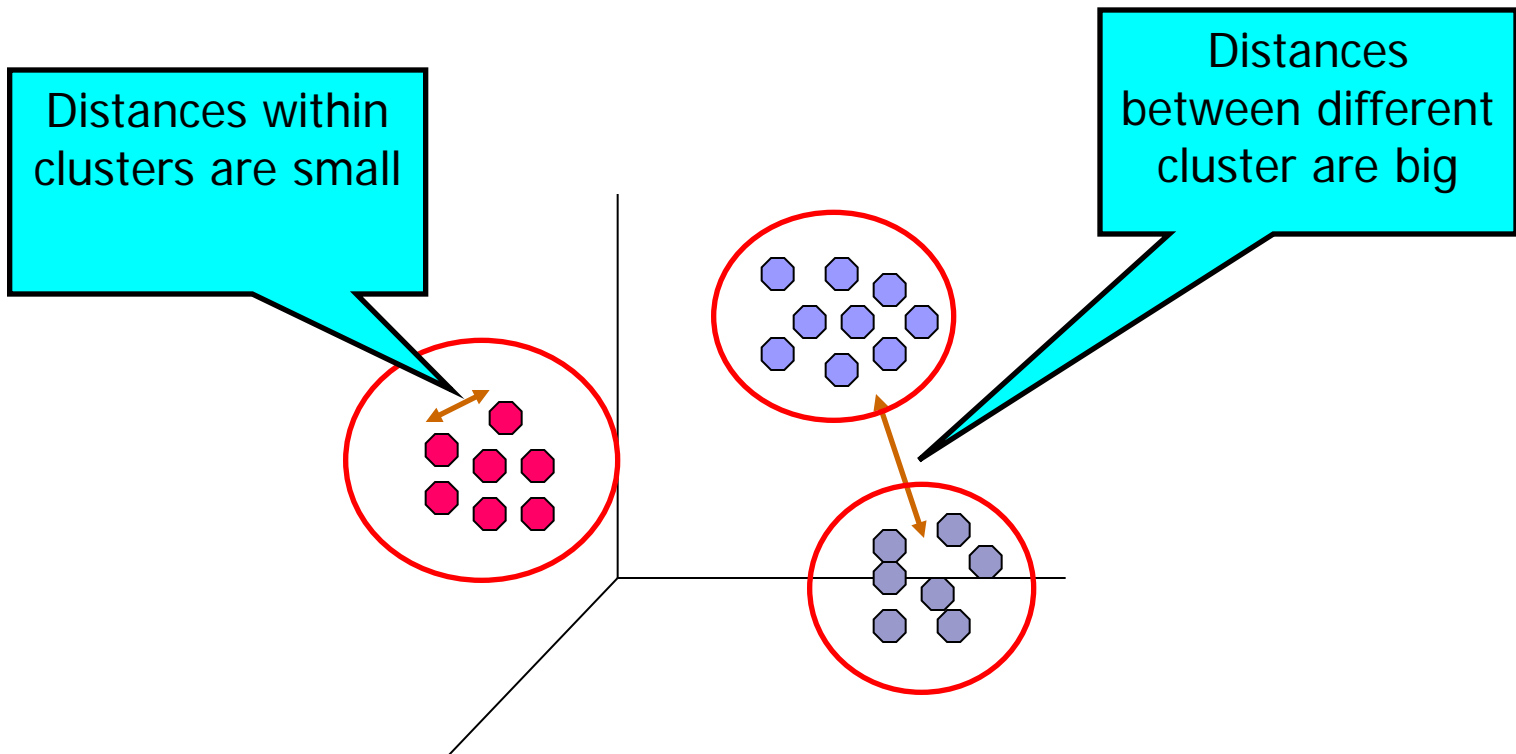
# Clustering Definition

- Common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data (Fayyad 96)
- Task that splits data into valid and usable groups (clusters), capturing the natural structure of the domain (Tan et al. 2004)



# Clustering Definition(I)

- **Cluster:** group or set of objects that:
  - Are similar to any other included in the same cluster .
  - Are not similar (different) to the objects in other clusters.



# Definición de clustering (II)

## Clustering:

- Split a heterogeneous population into a number of homogeneous subgroups called clusters.

## ■ Typical uses:

- As a preprocessing task part of a KDD process involving other data mining techniques.
- As a proper data mining technique.

# Formal Definition

Given a data base  $D=\{t_1, t_2, \dots, t_n\}$  of tuples and a value  $k$ , define a mapping  $f$  such as:

$$f: D \rightarrow \{1, \dots, k\}$$

Where each tuple  $t_i$  is assigned to one cluster  $K_j$ ,  $1 \leq j \leq k$ .

In order to perform the partition of the tuples in the database, a distance measure (or similarity measure) among tuples is needed

# Distance Measures

- City-Block:

$$d_{ij} = \sum_{k=1}^p W_k |x_{ik} - x_{jk}|$$

- Euclidean:

$$d_{ij} = \sqrt{\sum_{k=1}^p W_k (x_{ik} - x_{jk})^2}$$

- Minkowski:

$$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^p W_k (x_{ik} - x_{jk})^\lambda} \quad \lambda > 0$$

# Usual Problems

- Clustering techniques must face some problems when dealing with the tuples of the database.

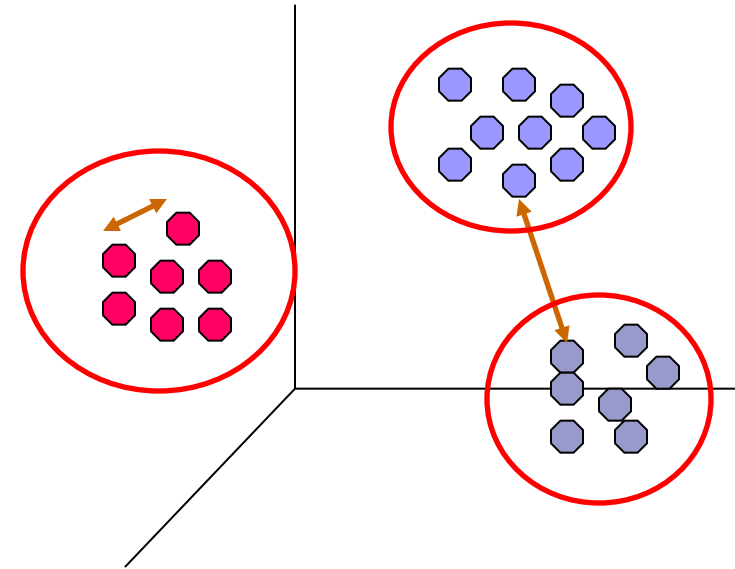
The most usual are:

- ☐ Measure units incompatibility. Scales
- ☐ Different type variables (numeric, symbolic, ordinal, time series, multivaluated, etc.).
- ☐ Missing values.
  - Consider as null
  - Mode, mean
  - Completely ignore the variable
- ☐ Noise.

# Evaluation of Clustering Techniques

■ A good clustering technique must output clusters that:

- Maximize intra-cluster similarity.
- Minimize inter-cluster similarity.



- The quality of the resulting clustering depends on the similarity measure and on its implementation.
- The quality of the clustering method is also measured based on its capacity to discover hidden patterns (one of the main achievements that a data mining technique can deliver).



# Desirable Properties

- Scalability.
- Deal with different types of variables.
- Discover clusters with arbitrary shapes.
- Deal with noisy and atypical data.
- Transparent to the order data is processed.
- Usable with high dimension data (number of tuples and/or number of attributes).
- Interpretable results.

# Applications

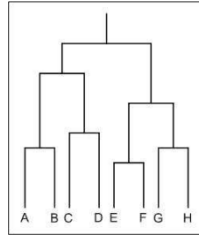
- Marketing: discover groups of clients with same needs or preferences given a large database of customer data containing their properties and past buying records. Use this knowledge for e-marketing, create new offers, etc.
- Insurance: identifying groups of motor insurance policy holders with similar claim costs. Identify frauds.
- Urban Planification: Identify neighbourhoods with similar houses according to their type, valor or geographical situation.
- WWW: Document classification, clustering weblog data to discover groups of similar access patterns
- Biology, Libraries, Sismology, etc.





# III: Clustering Algorithms

# Clustering Algorithms types



- Hierarchical
- Partitional
- Density Based
- Restriction Based
- Grid
- Others (Coclustering, graph-partitioning)



# Clustering Algorithms properties

- Agglomerative vs Divisive
- Monothetic vs Polythetic
- Hard vs Fuzzy
- Deterministic vs Stochastic
- Incremental vs Non.incremental



# Hierarchical Clustering

- Generate levels of clusters (hierarchies) with ordered levels of aggregation/division.
- Hierarchies are usually binary trees called dendograms.
- Types:
  - Agglomerative
  - Divisive



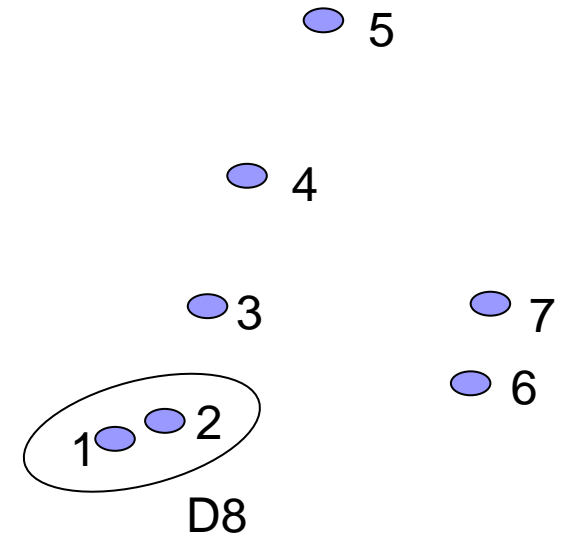
# Agglomerative Hierarchical Clustering

1. Assign each object to its own cluster. Construct the list of inter-cluster distances and sort it.
2. Locate the pair of cluster most similar and group them into a single cluster.
3. Compute the similarity between the new cluster and the rest:
  - a) Single-Link.
  - b) Complete-Link.
  - c) Average-Link.
4. Repeat steps 2 and 3 until all objects are included in one cluster

# Agglomerative Hierarchical Clustering: Example (I)

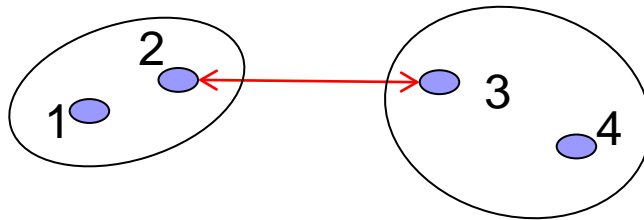
- Step 1: Compute the similarity matrix or distance matrix.
- Locate the closest two clusters (objects 1 and 2).

D1							
D2	0,3606						
D3	0,5	0,4243					
D4	0,9220	0,7071	0,4472				
D5	1,3416	1,0440	0,9220	0,5000			
D6	1,8385	1,5524	1,3892	0,9434	0,5099		
D7	1,7263	1,5000	1,2369	0,8062	0,5831	0,4000	
D1	D2	D3	D4	D5	D6	D7	

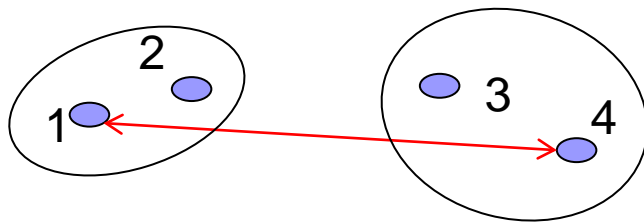


# How to apply distances

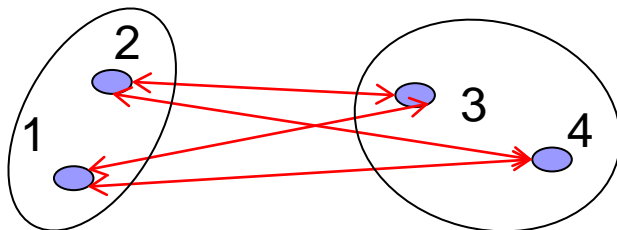
- Single link (Enlace Simple )



- Complete link (Enlace Completo)



- Average link (Enlace Promedio)



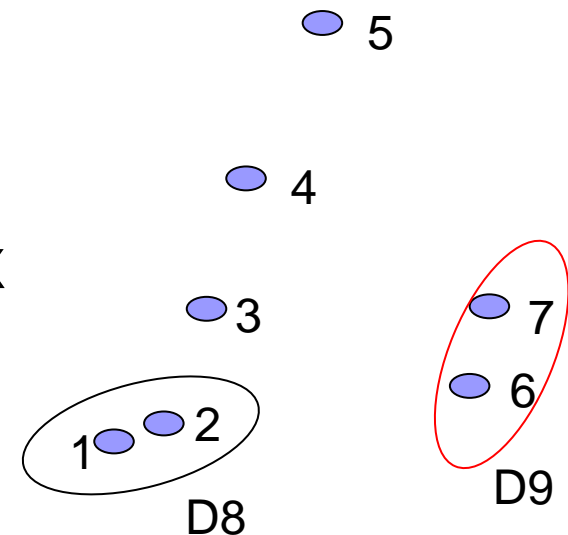
# Agg. Hier. Clustering: Ex. (II)

- Merge objects 1 and 2 into a single cluster D8.

D1							
D2	0,3606						
D3	0,5000	0,4243					
D4	0,9220	0,7071	0,4472				
D5	1,3416	1,0440	0,9220	0,5000			
D6	1,8385	1,5524	1,3892	0,9434	0,5099		
D7	1,7263	1,5000	1,2369	0,8062	0,5831	0,4000	
D1	D2	D3	D4	D5	D6	D7	

- Compute the new Similarity Matrix

D1/D2=D8						
D3	0,4243					
D4	0,7071	0,4472				
D5	1,0440	0,9220	0,5000			
D6	1,5524	1,3892	0,9434	0,5099		
D7	1,5000	1,2369	0,8062	0,5831	0,4000	
D1/D2=D8	D3	D4	D5	D6	D7	



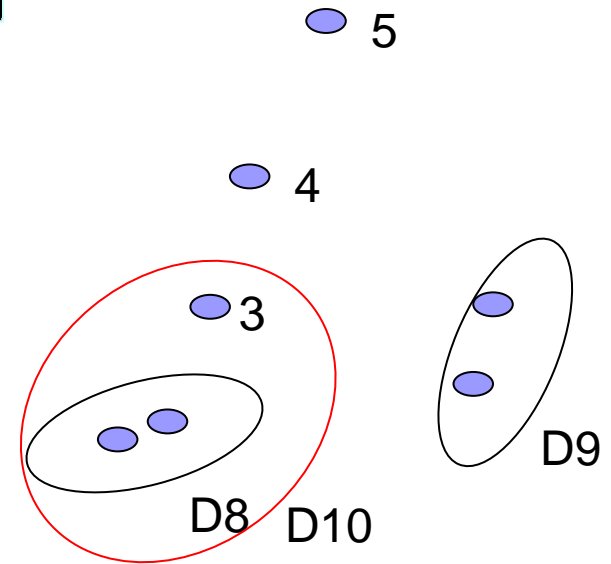


# Agglomerative Hierarchical Clustering: Example (III)

D1/D2=D8						
D3	0.4243					
D4	0.7071	0.4472				
D5	1.3416	0.9220	0.5000			
D6	1.5524	1.3892	0.9434	0.5099		
D7	1.5000	1.2369	0.8062	0.5831	0.4000	
D1/D2=D8	D3	D4	D5	D6	D7	



D8					
D3	0.4243				
D4	0.7071	0.4472			
D5	1.3416	0.9220	0.5000		
D6/D7=D9	1.5000	1.2369	0.8062	0.5099	
D8	D3	D4	D5	D6/D7=D9	

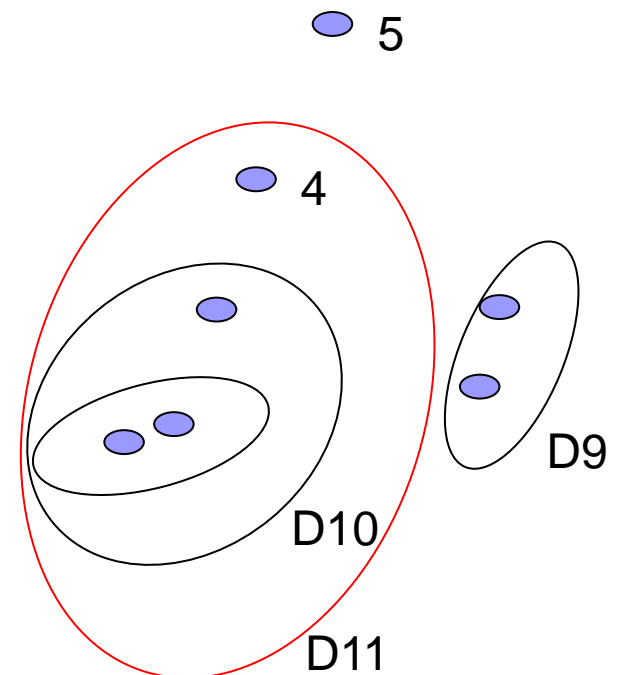


# Agglomerative Hierarchical Clustering: Example (IV)

D8				
D3	0.4243			
D4	0.7071	0.4472		
D5	1.3416	0.9220	0.5000	
D6/D7=D9	1.5000	1.2369	0.8062	0.5831
D8	D3	D4	D5	D6/D7=D9



D3/D8=D10				
D4	0.4472			
D5	0.922	0.5		
D9	1.2369	0.8062	0.5831	
D3/D8=D10	D4	D5	D9	



# Agglomerative Hierarchical Clustering: Example (V)

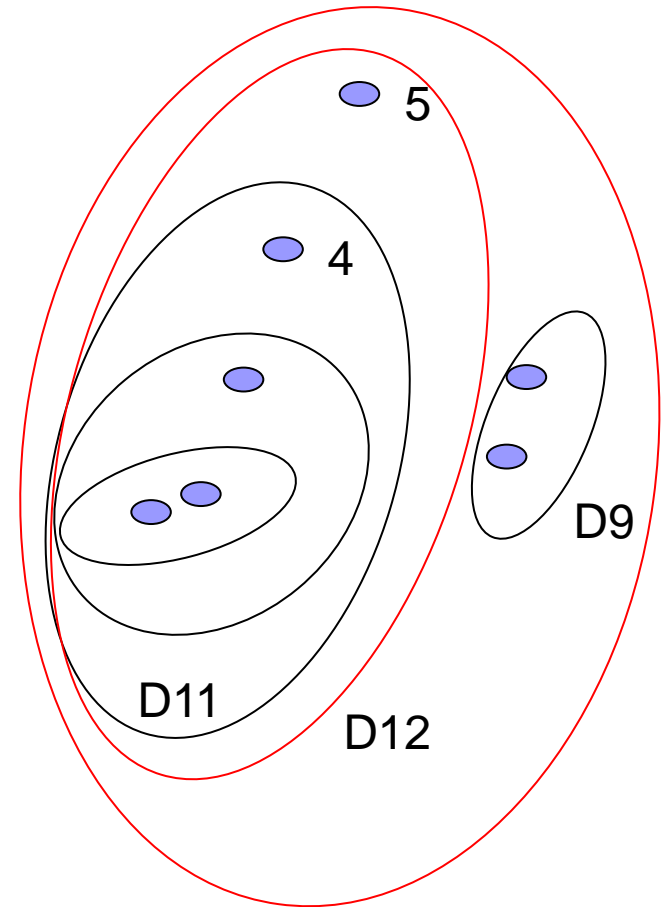
<b>D3/D8=D10</b>				
D4	0,4472			
D5	0,922	0,5		
D9	1,2369	0,8062	0,5831	
	<b>D3/D8=D10</b>	<b>D4</b>	<b>D5</b>	<b>D9</b>



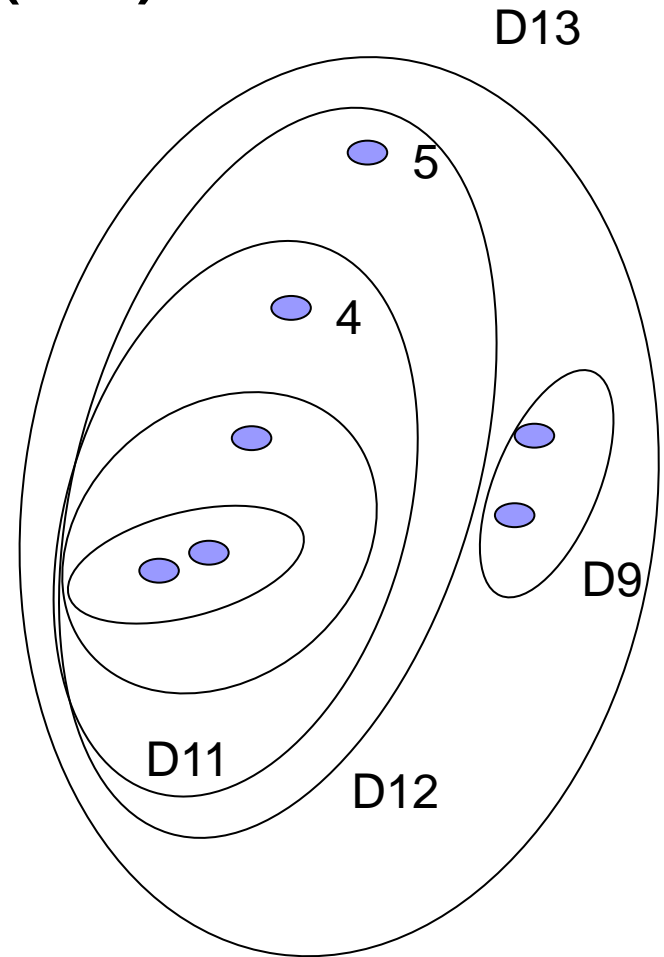
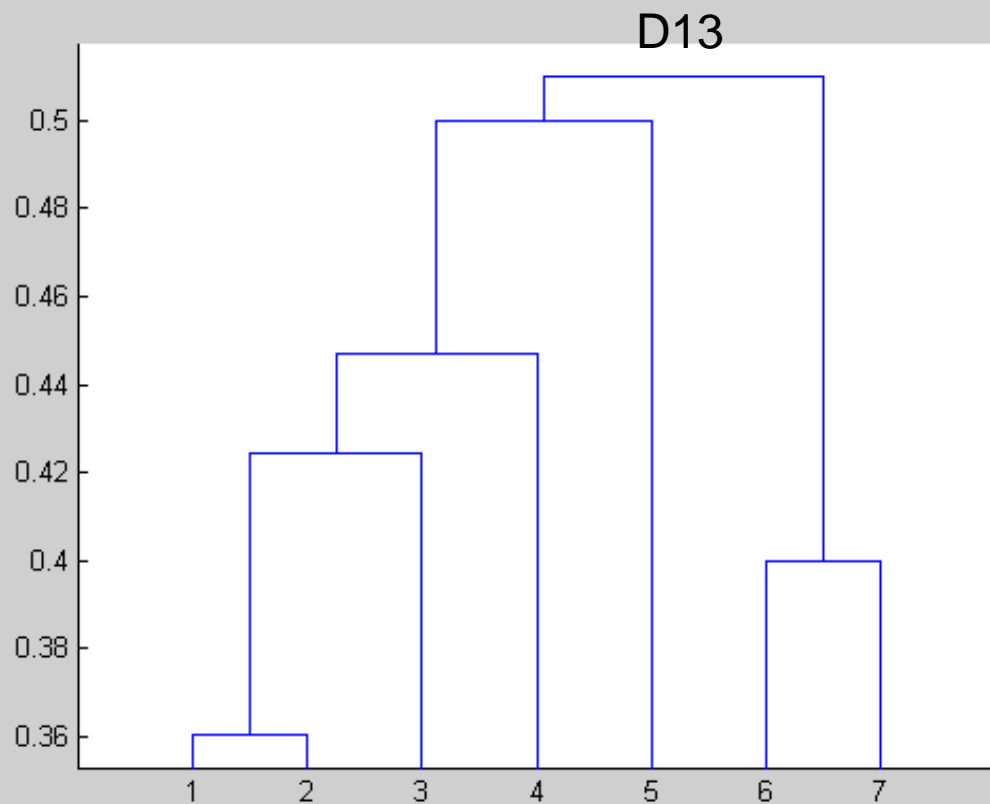
<b>D4/D10=D11</b>			
D5	0,5000		
D9	0,8062	0,5831	
	<b>D4/D10=D11</b>	<b>D5</b>	<b>D9</b>



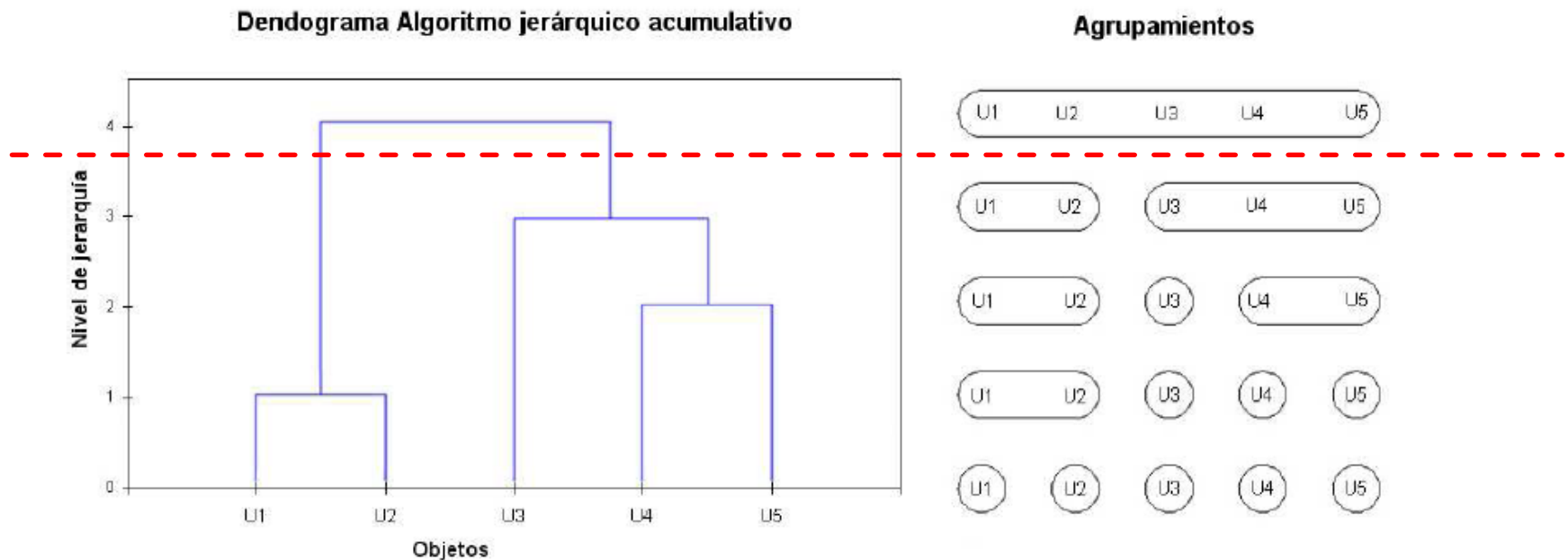
<b>D5/D11=D12</b>		
D9	0,5831	
	<b>D5/D11</b>	<b>D9</b>



# Agglomerative Hierarchical Clustering: Example (VI)



# Hierarchical Clustering



$\text{Distancia-Max-Nodo} / N^{\circ} \text{ Atributos} < \text{Umbral}$

Distancia-Max-Nodo = mayor distancia entre dos objetos del nodo  
Umbral definido por el experto



# Divisive Hierarchical Clustering

Start with all objects in one cluster and iteratively split the clusters into smaller ones

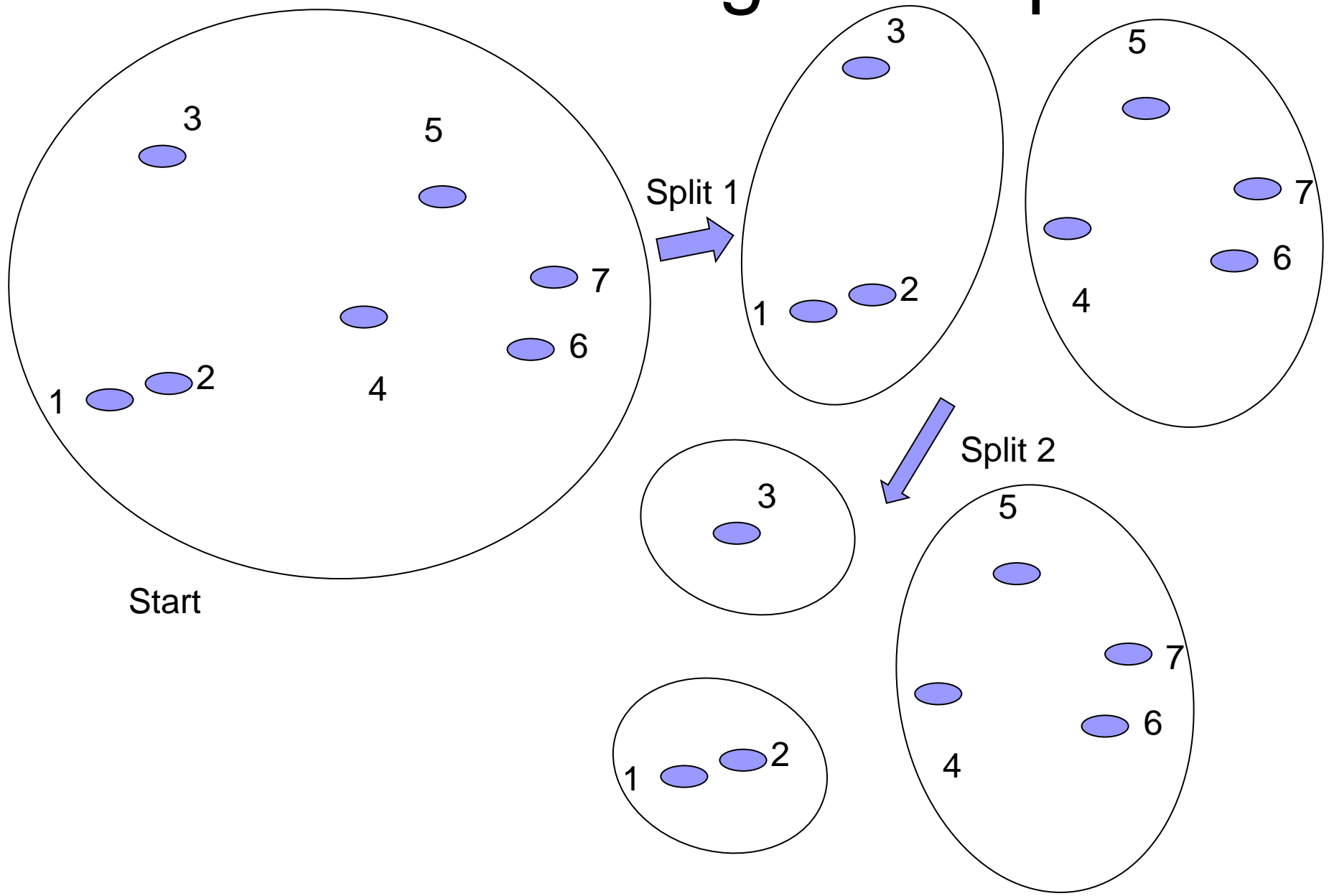
- Method is more complex (needs a splitting clustering algorithm)
- Can be more efficient than agglomerative approaches
- Divisive algorithms produce more accurate hierarchies than bottom-up algorithms in some circumstances



# Divisive Hierarchical Clustering

1. Start at the top with all objects in one cluster.
2. Select the next cluster to split based on some criterion
3. Split the chosen cluster into sub-clusters
  1. How many subclusters?
  2. Choose a flat clustering algorithm
4. Repeat steps 2 and 3 until all objects are included in its own singleton cluster

# Divisive Clustering: Example





# Step 2: choosing the cluster to split

Given:

- $U = \{v_1, \dots, v_n\}$ , the set of all objects
- A partitioning clustering  $U = \{C_1, C_2, \dots, C_k\}$

Define:

$\text{cutcost}(C_p) = \sum \text{sim}(v_i, v_j). v_i \text{ in } C_p, v_j \text{ in } U - C_p$  *External Sim.*

$\text{intracost}(C_p) = \sum \text{sim}(v_i, v_j). v_i, v_j \text{ in } C_p$  *Internal Sim.*

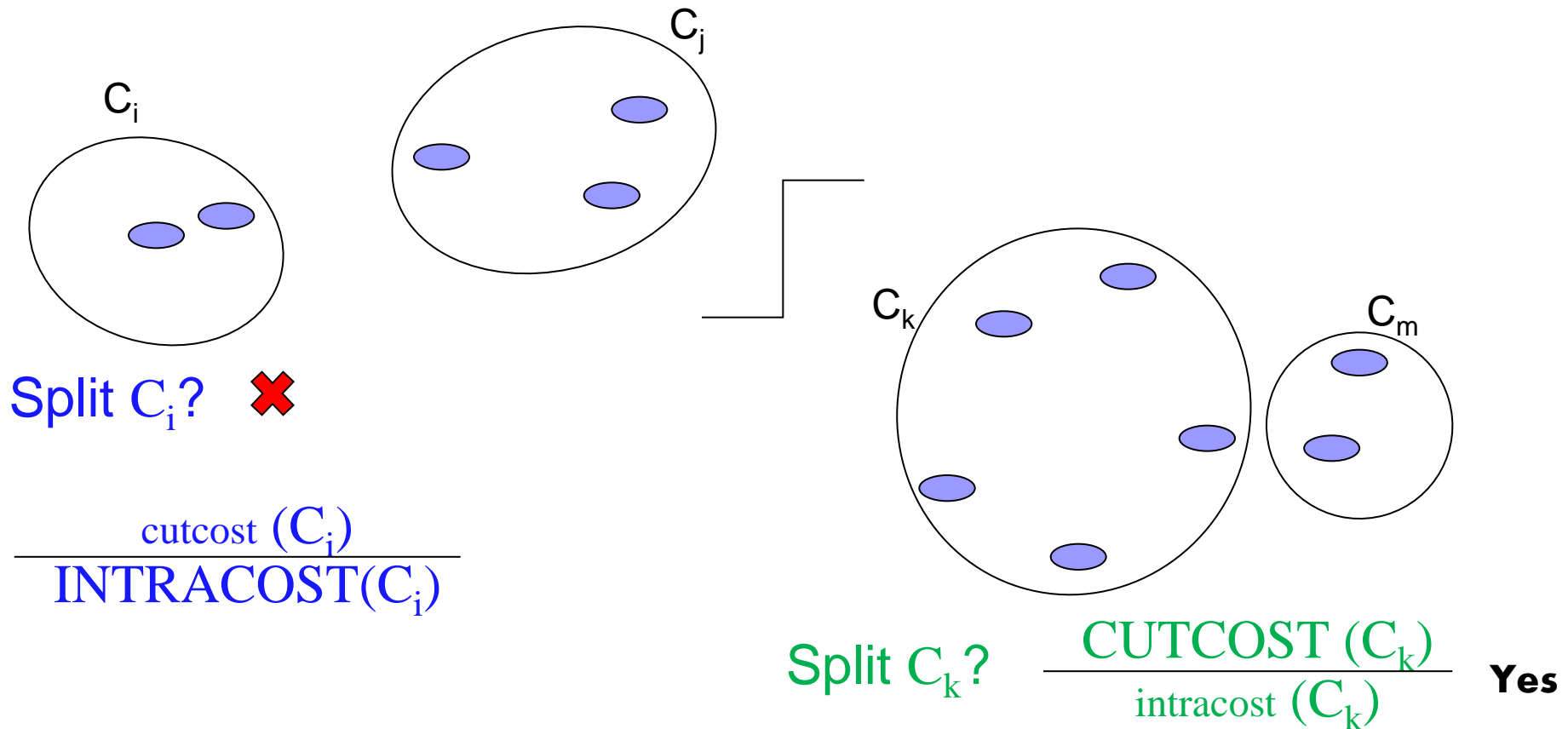
- $\text{cost}(C_1, \dots, C_k) = \sum \text{cutcost}(C_p) / \text{intracost}(C_p)$

## min-max cut optimization

Find clustering  $C_1, \dots, C_k$  that minimizes

$\text{cost}(C_1, \dots, C_k)$

# Choosing the cluster to split





# Hierarchical Clustering

## ■ Advantages:

- ☐ Commonly used.
- ☐ Available in statistics and DM tools.
- ☐ Do not need initialization parameters.
- ☐ Very efficient.

## ■ Some Implementations:

- ☐ BIRCH, CURE, CHAMALEON.

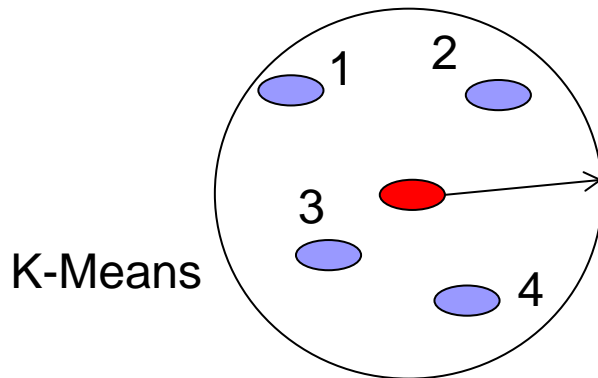


# Partitional Clustering

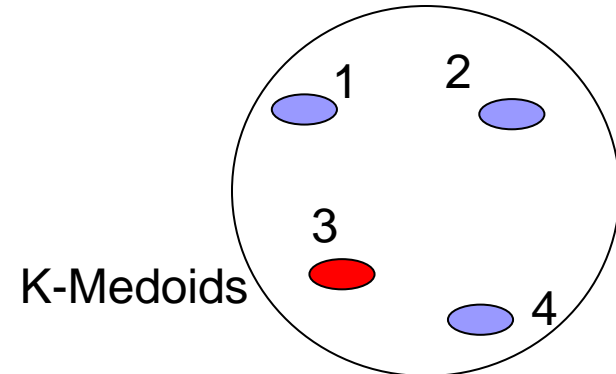
- An initial split is done, usually randomly.
- Each iteration reassigns objects to different clusters, achieving the optimal partition after some iterations.
- The desired number of clusters is a parameter.
- Most used partitional algorithms:
  - K-means.
  - K-medoids.

# Partitional Clustering K-means:

1. Select the initial  $k$  centroids (randomly).
2. Assign each object to the cluster of the closest centroid.
3. Recalculate the centroids of each cluster.
  - K-means  $\rightarrow$  Central object of the cluster
4. Repeat step 2 until a criterium of convergence is met (none or little reassignments between clusters, threshold of intra/inter-cluster similarity)).



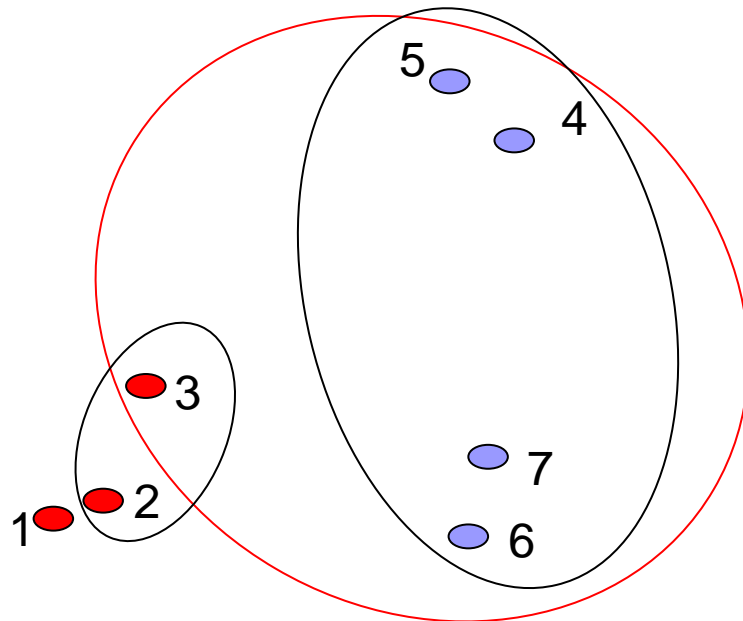
*Objeto más centrado en el cluster  $\rightarrow$  K-medoids*



# Partitional Clustering K-means:

## ■ How to select the initial centroids:

- First  $k$  objects.
- *Randomly select  $k$  objects.*
- Randomly distribute all objects into  $k$  clusters and take the center of those clusters.



Initial cluster centers 1,2,3 →

$C1=\{1\}$ ,  $C2=\{2\}$ ,  $C3=\{3,4,5,6,7\}$

Final clusters:

$C1=\{1\}$ ,  $C2=\{2,3\}$ ,  $C3=\{4,5,6,7\}$


# K-means: Example (I)

	at1	at2	K=2	centrodes			
Obj1	0,8	1,8		c1	0,8	1,8	d1
Obj2	1,1	1,6		c2	1,1	1,6	d2
Obj3	0,8	1,3					
Obj4	1	0,9					
Obj5	1,4	0,6					
Obj6	1,5	0,1	→				
Obj7	1,1	0,1	→				
Obj8	0,5	2,4					
Obj9	1,5	0,4					

8



1



2



3



4




7




5



9

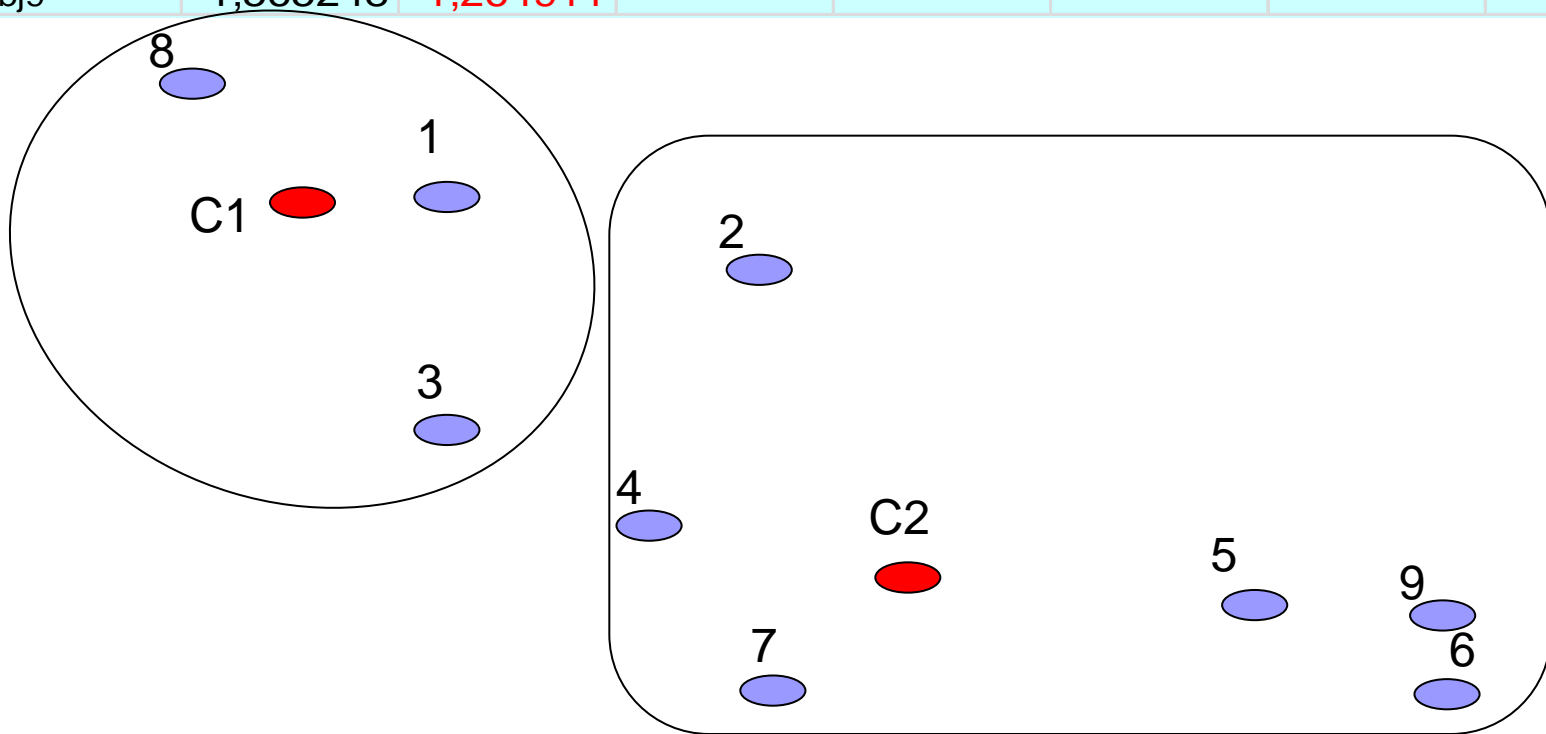


6



# K-means: Example (II)

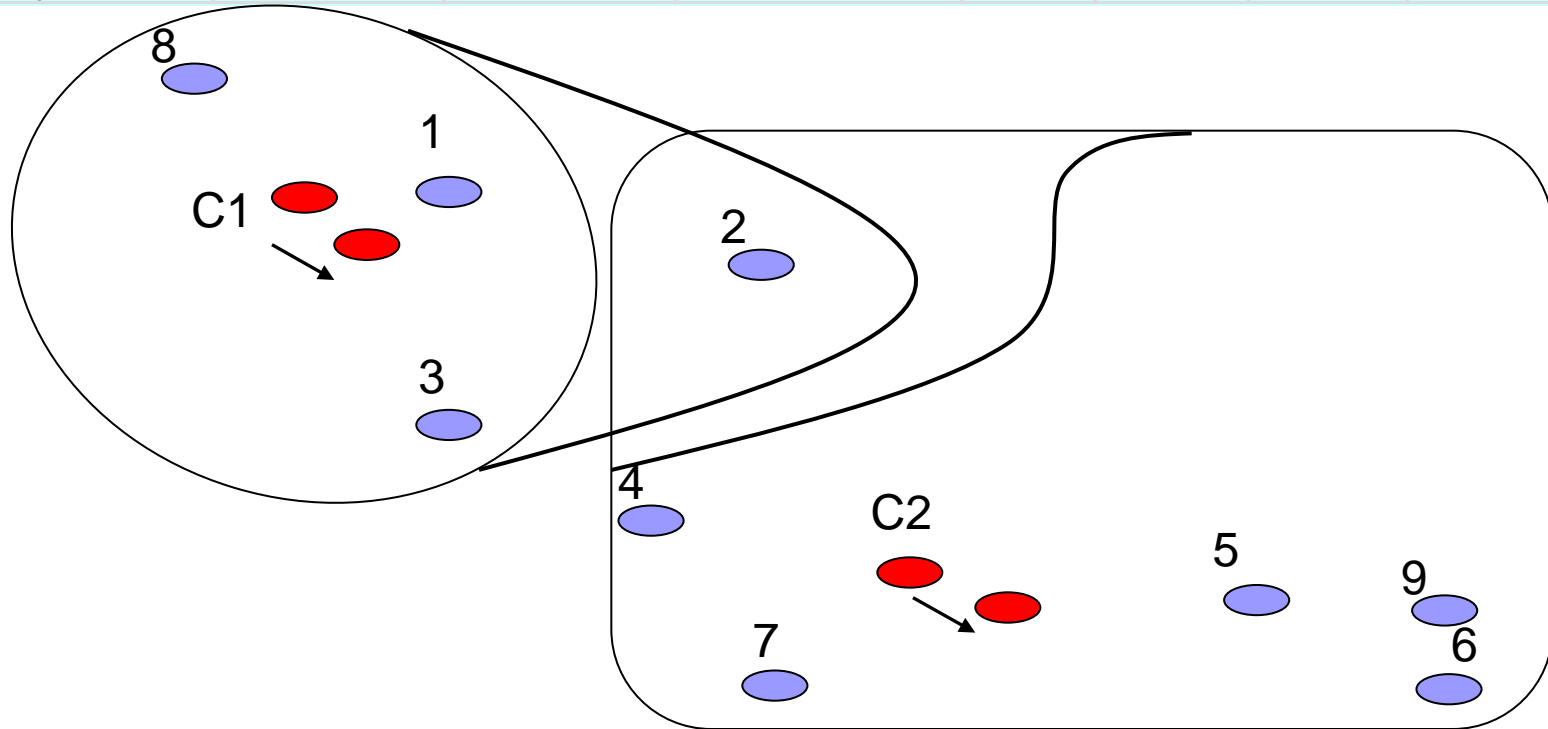
distances to centroids							
	c1	c2					Ndatos
Obj1	0	0,360555	iter 1	c1	0,7	1,83333333	d1,d3,d8
Obj2	0,360555	0		c2	1,26666667	0,61666667	d2,d4,d5,d6,d
Obj3	0,4	0,424264					
Obj4	0,921954	0,707107					
Obj5	1,341641	1,044031					
Obj6	1,838478	1,552417					
Obj7	1,726268	1,5	→				
Obj8	0,67082	1	→				
Obj9	1,565248	1,264911					





# K-medias: Example (III)

	c1	c2	ITERATION 2	c1	0,8	1,775	d1,d2,d3,d8 d4,d5,d6,d7, d9
Obj1	<b>0,105409</b>	1,272028		c2	1,3	0,42	
Obj2	<b>0,463081</b>	0,997358					
Obj3	<b>0,542627</b>	0,827479					
Obj4	0,980363	<b>0,389087</b>					
Obj5	1,418136	<b>0,134371</b>					
Obj6	1,909043	<b>0,566912</b>					
Obj7	1,778889	<b>0,542883</b>	→				
Obj8	<b>0,600925</b>	1,941148	→				
Obj9	1,641476	<b>0,318416</b>					





# Partitional Clustering

## ■ Advantages:

- ☐ Widely used.
- ☐ Many implementations and documentation.
- ☐ Efficient.
- ☐ Simple.

## ■ Disadvantages:

- ☐ Requires the number of clusters  $k$  as initial parameter.
- ☐ Very susceptible to noisy data.
- ☐ Highly dependent on the initial random partition

# Density based Clustering

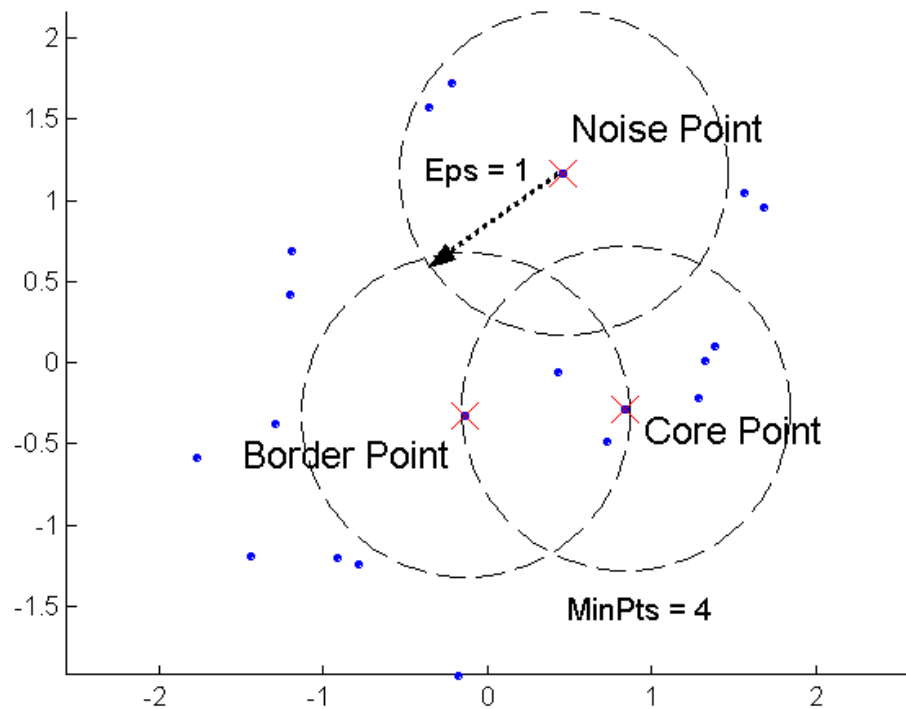
- Clustering methods based on the concept of density around each object: number of objects close enough to that object (predefined radius).
- Allows clusters with irregular shapes.
- DBSCAN.
- Parameters:
  - *Eps*: radius to compute neighborhood .
  - *MinPts*: minimum number of objects in the neighborhood of the object in order to be considered *dense*.

# Density based Clustering

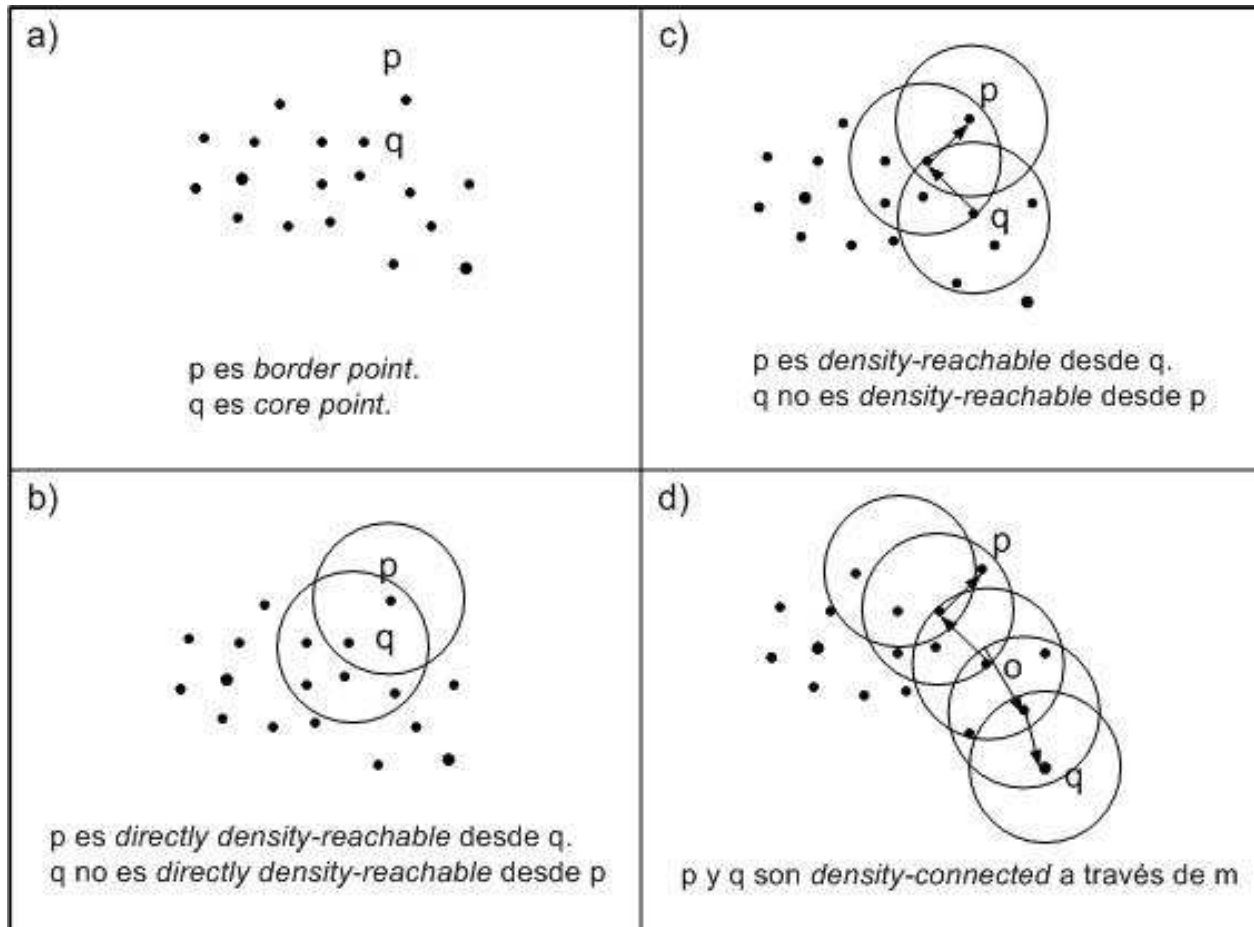
## ■ Classification of Objects:

- Core: objects that have more than  $MinPts$  neighbours within their vicinity ( $Eps$ ).
- Border: objects that have less than  $MinPts$  neighbours within their vicinity ( $Eps$ ), but are in the neighborhood of a core point.
- Noise: Objects that are nor Core nor Border objects.

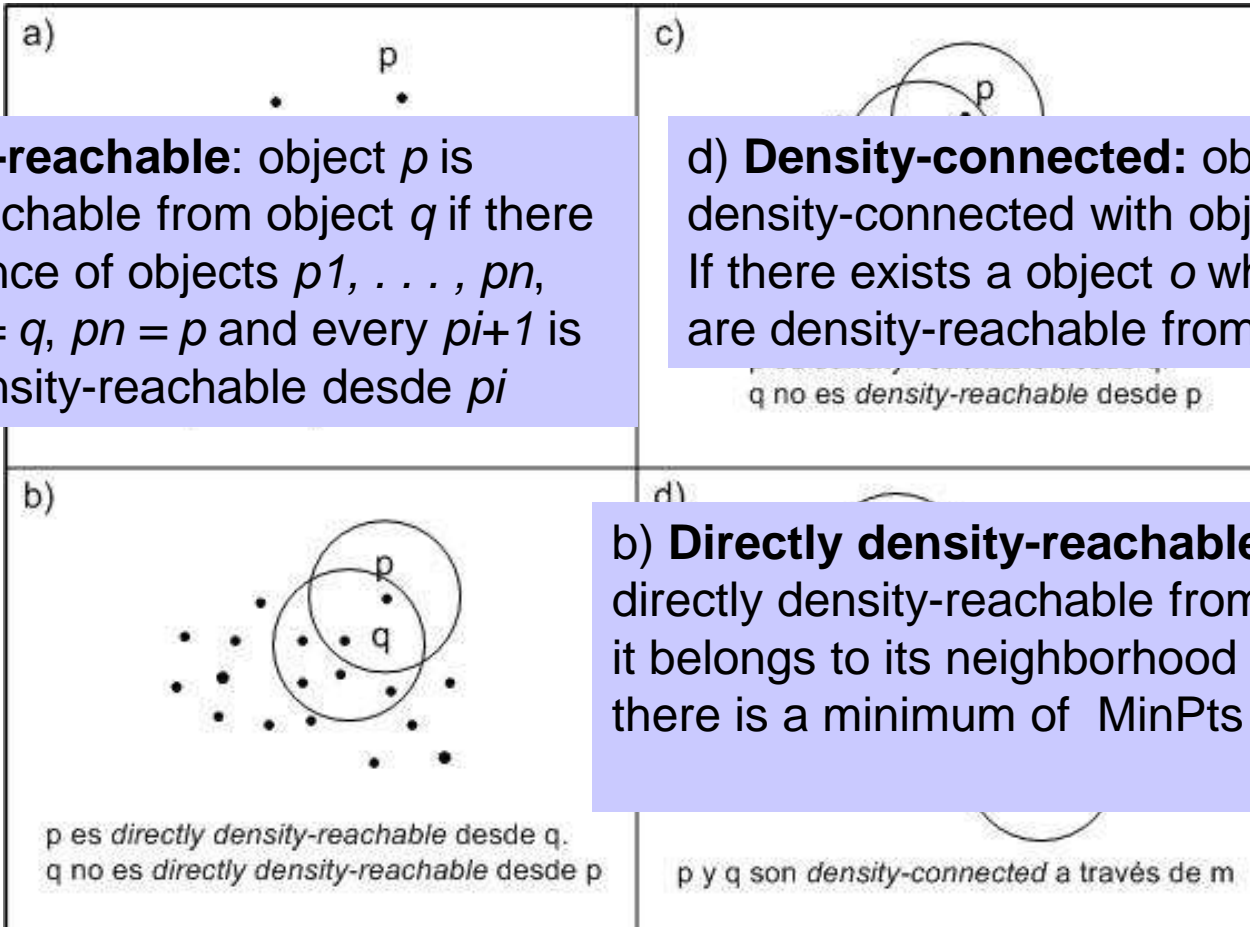
# Density based Clustering



# DBSCAN Concepts



# DBSCAN Concepts



c) **Density-reachable:** object  $p$  is density-reachable from object  $q$  if there is a sequence of objects  $p_1, \dots, p_n$ , where  $p_1 = q$ ,  $p_n = p$  and every  $p_{i+1}$  is directly density-reachable desde  $p_i$

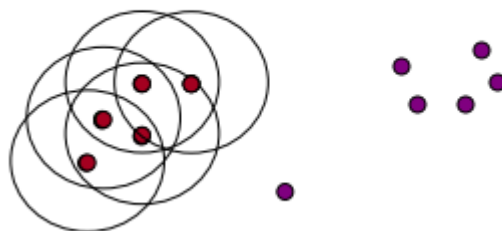
d) **Density-connected:** object  $p$  is density-connected with object  $q$  if there exists a object  $o$  where  $p$  and  $q$  are density-reachable from  $h$

b) **Directly density-reachable:** object  $p$  is directly density-reachable from object  $q$  if it belongs to its neighborhood  $Eps$  and there is a minimum of  $MinPts$  in it.

# How does DBSCAN works?

(Jing Gao – State Univ. NY Buffalo)

- radio  $Eps = 2$  cm
- MinPts = 3



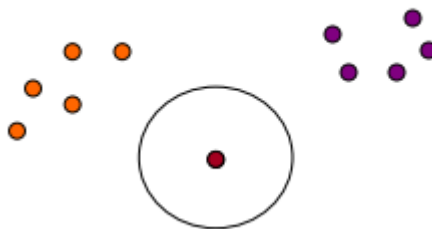
```
For EACH point  $p$  DO
  IF  $p$  NOT yet classified THEN
    IF  $p$  is core point THEN
      Create a new cluster with all density-reachable
        points from  $p$ 
    ELSE classify  $p$  as NOISE
```



# How does DBSCAN works?

(Jing Gao - State Univ. NY Buffalo)

- radio  $Eps = 2$  cm
- MinPts = 3

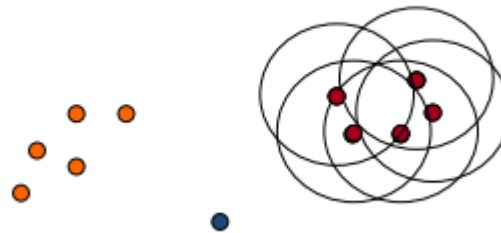


```
For EACH point  $p$  DO
  IF  $p$  NOT yet classified THEN
    IF  $p$  is core point THEN
      Create a new cluster with all density-reachable
        points from  $p$ 
    ELSE classify  $p$  as NOISE
```

# Ejemplo BSCAN

(Jing Gao - State Univ. NY Buffalo)

- radio  $Eps = 2$  cm
- MinPts = 3



```
For EACH point  $p$  DO
  IF  $p$  NOT yet classified THEN
    IF  $p$  is core point THEN
      Create a new cluster with all density-reachable
        points from  $p$ 
    ELSE classify  $p$  as NOISE
```

## DBSCAN - A Density-Based Algorithm for Discovering Clusters in Large Spatial Database Noise Data.

DBSCAN(D, eps, MinPts)

  C = 0 for each unvisited point P in dataset D mark P as visited

  N = regionQuery(P, eps)

  if sizeof(N) < MinPts mark P as NOISE

  else C = next cluster

    expandCluster(P, N, C, eps, MinPts)

expandCluster(P, N, C, eps, MinPts)

  add P to cluster C

  for each point P' in N

  if P' is not visited

    mark P' as visited

    N' = regionQuery(P', eps)

    if sizeof(N') >= MinPts N = N joined with N'

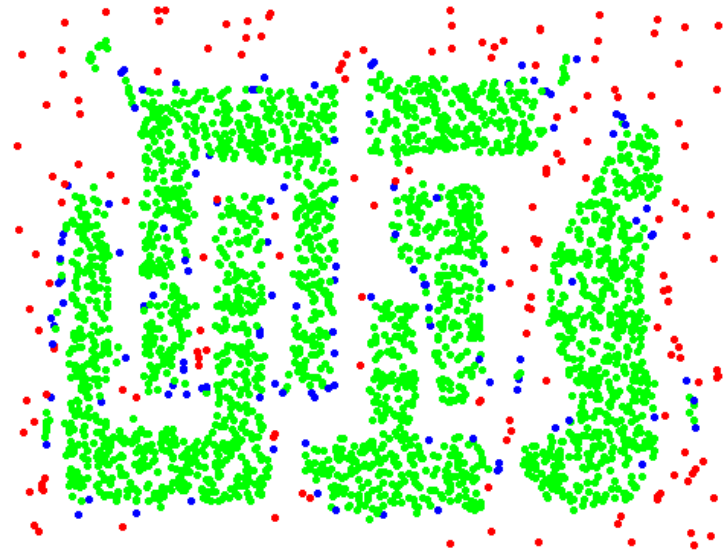
    if P' is not yet member of any cluster

      add P' to cluster C

# Density based Clustering



Initial objects



Classification according to Core,  
Border and Noise objects

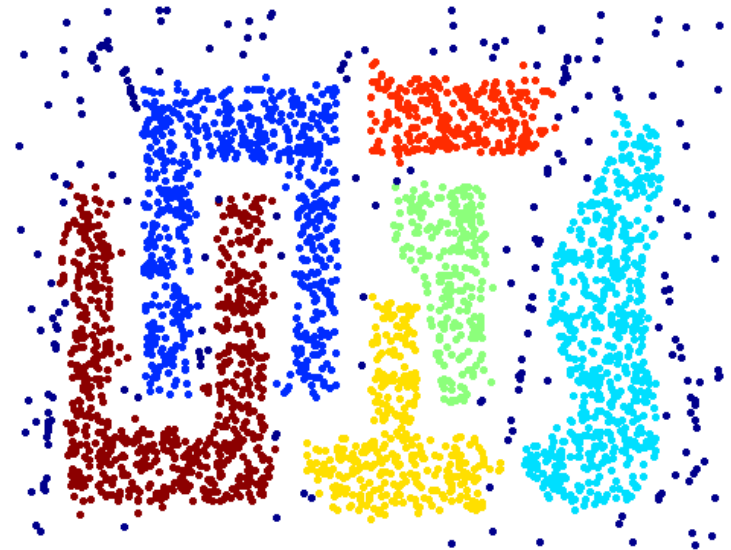
Eps = 10, MinPts = 4

# Density based Clustering

DBSCAN:



Initial objects



Clusters

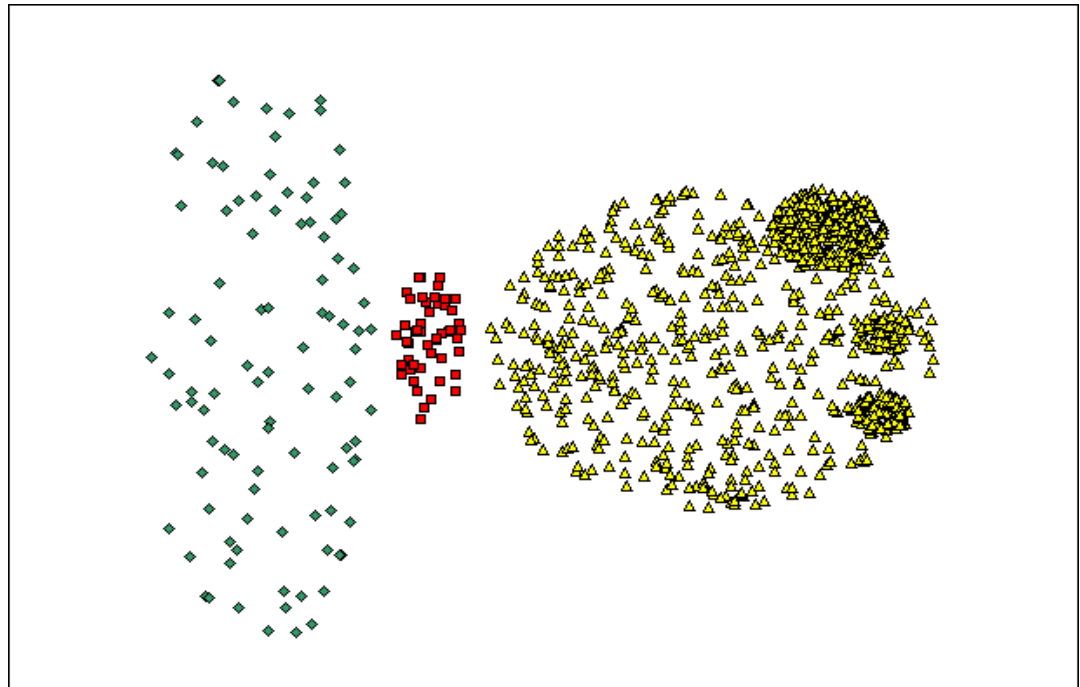
# Density based Clustering

- Advantages:

- ☐ Works well with noisy data.
- ☐ Can produce clusters with different shapes and size.

- Disadvantages:

- ☐ Efficiency problems with high dimensional data.
- ☐ Performs badly with data with different density regions.



# Semi-supervised Clustering

- In many cases a small amount is available concerning either:
  - Pairwise constraints: must-link, cannot-link
  - Class labels for some items
- In the previous methods, this knowledge can be used only for validation purposes

But, can it be used in a more interesting way, to guide the clustering process?



# Semi-supervised Clustering

- Pairwise y Class labels constraints are equivalent
- How to make the constraints and the clustering distance consistent:
  - Similarity adapting methods use an existing clustering method, adapting the similarity measure
  - Search-Based methods use the constraints to guide the clustering (initialize clusters, cost function with penalty based constraints, etc.)





# Grid Clustering

- Define the grid
- Compute significant cells
- Create clusters based on adjacent significant cells
- Redefine the grid, iterate and compare clusters



# Part IV: Conclusions



# Conclusions

- KDD → Data Mining → Descriptive problems → Clustering.
- Split a population group into homogenous groups.
- Algorithms:
  - Hierarchical Clustering.
  - Partitional Clustering.
  - Density-based Clustering.
- Applications to almost every domain or problem.

# Bibliography

- CHAMALEON: Karypis, G., Han, E. H., Kumar, V., Chameleon: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer, 32(8), pp. 68-75, (1999)
- BIRCH: Zhang, T., Ramakrishnan, R., Livny, M., BIRCH: An efficient data Clustering method for very large databases. Proceedings of ACM SIGMOD Conference on Management of Data, pp. 103-114, (1996)
- CURE: Guha, S., Rastogi, R., Shim, K., CURE: An efficient clustering algorithm for large databases. Proceedings of ACM SIGMOD'98, pp. 73-84, (1998)
- K-MEANS: MacQueen, J. B., Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1, pp. 281-297, (1987)
- K-MEDOIDS: Kaufman, L., Rousseeuw, P.J., Clustering by means of medoids. Y. Dodge Ed., Statistical Data Analysis based on the L1 Norm (North-Holland, Amsterdam, pp. 405-416, (1987)
- DBSCAN: Ester, M., Kriegel, H.P., Sander, J., Xu, X., A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD'96), pp. 226-231, (1996)