

Clustering

part II

Clustering

- What is Cluster Analysis?
- Types of Data in Cluster Analysis
- A Categorization of Major Clustering Methods
- **Partitioning Methods**
- Hierarchical Methods

Partitioning Algorithms: Basic Concept

- Partitioning method: Construct a partition of a database D of n objects into a set of k clusters
- Given a k , find a partition of k clusters that optimizes the chosen partitioning criterion
 - **Global optimal**: exhaustively enumerate all partitions
 - **Heuristic methods**: k-means and k-medoids algorithms
 - **k-means**: Each cluster is represented by the center of the cluster
 - **k-medoids or PAM** (Partition around medoids): Each cluster is represented by one of the objects in the cluster

k-means variations

- *k*-medians clustering
 - instead of mean, use medians of each cluster
 - not affected by extreme values
 - works like *k*-means
- *k*-medoids
 - instead of mean use an actual data point as a cluster centroid
 - PAM algorithm

The *K-Medoids* Clustering Method

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
 - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling

PAM (Partitioning Around Medoids)

- PAM (Kaufman and Rousseeuw, 1987)
- Use real objects to represent clusters
 - Select k representative objects arbitrarily
 - For each medoid m
 - For each non-medoid O
 - Swap m and O and compute cost of resulting configuration
 - Select configuration with the lowest cost
 - Repeat until there is no change

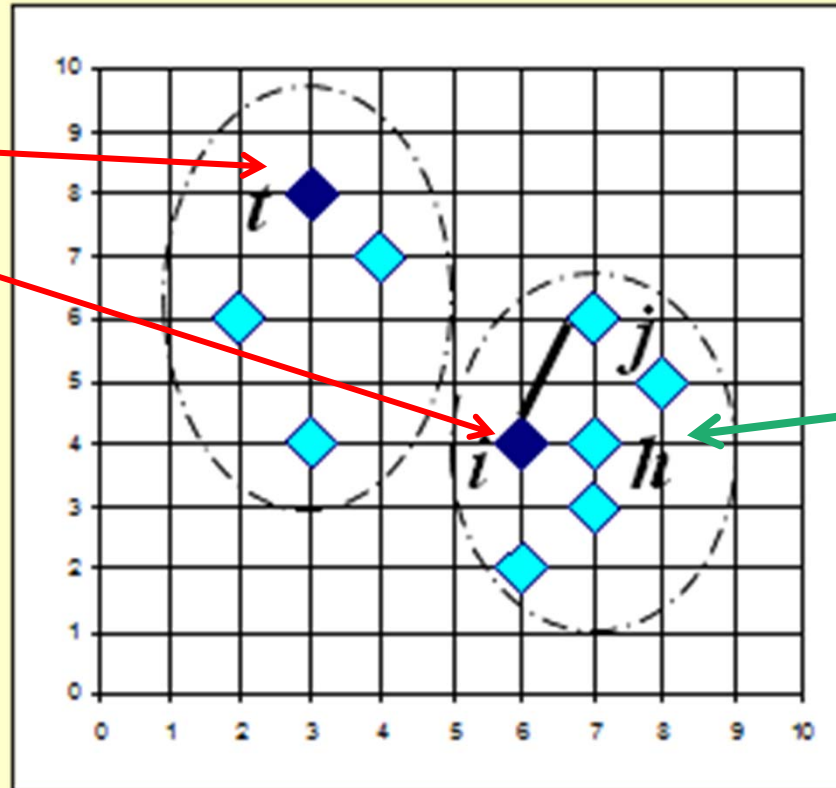
PAM Clustering: Total swapping cost $TC_{ih} = \sum_j C_{jih}$

i, t : medoids

h : medoid candidate

j : a point

C_{jih} : swapping cost due to j



If the medoid i is replaced by h as the cluster representative the cost associated to j is

$$C_{jih} = d(j, h) - d(j, i)$$

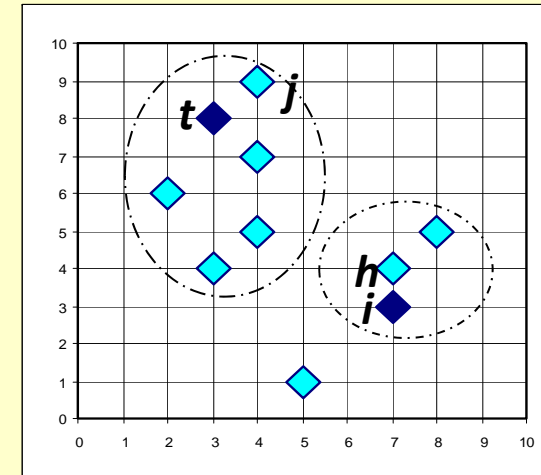
PAM Clustering: Total swapping cost $TC_{ih} = \sum_j C_{jih}$

examples of other scenarios

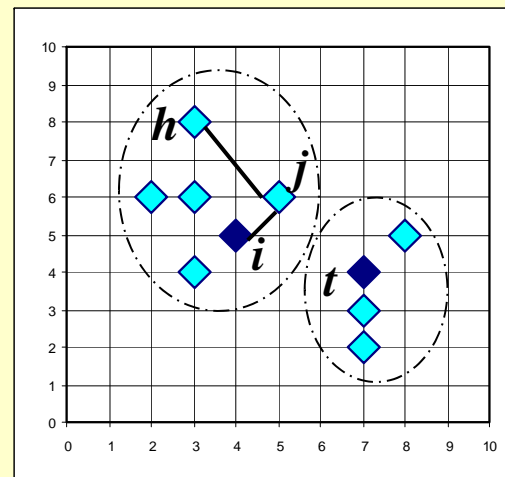
i, t : medoids

h : medoid candidate

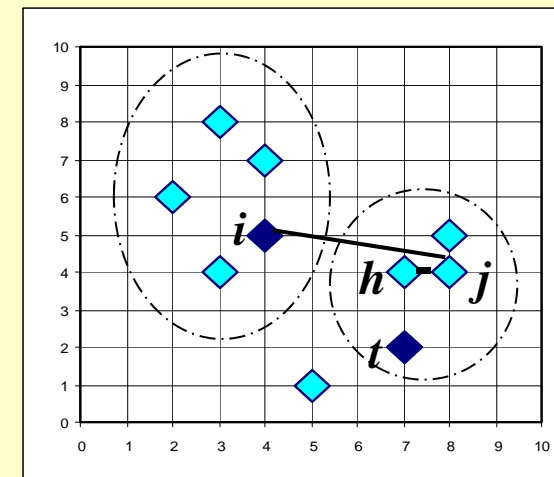
there is no cost induced by j



$$C_{jih} = 0$$



$$C_{jih} = d(j, t) - d(j, i)$$



$$C_{jih} = d(j, h) - d(j, t)$$

Comments on PAM

- Pam is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- *PAM* works effectively for small data sets, but does not scale well for large data sets

CLARA (Clustering LARge Applications)

- CLARA (Kaufmann and Rousseeuw in 1990) **draws a sample of the dataset** and applies PAM on the sample in order to find the medoids.
- If the sample is representative the medoids of the sample should approximate the medoids of the entire dataset.
- Medoids are chosen from the sample. Note that the algorithm cannot find the best solution if one of the best k-medoids is not among the selected sample.
- To improve the approximation, **multiple samples are drawn** and the best clustering is returned as the output
- The clustering accuracy is measured by the **average dissimilarity** of all objects in the entire dataset.
 - Experiments show that 5 samples of size $40+2k$ give satisfactory results

CLARA Algorithm

- For $i = 1$ to 5, repeat the following steps
- Draw a sample of $40 + 2k$ objects randomly from the entire data set, and call Algorithm PAM to find k medoids of the sample.
- For each object in the entire data set, determine which of the k medoids is the most similar to it.
- Calculate the average dissimilarity ON THE ENTIRE DATASET of the clustering obtained in the previous step. If this value is less than the current minimum, use this value as the current minimum, and retain the k medoids found in Step (2) as the best set of medoids obtained so far.
- Return to the first step and repeat.

CLARA (Clustering LARge Applications)

- Strengths and Weaknesses:
 - Deals with larger data sets than *PAM*
 - Efficiency depends on the sample size
 - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

CLARANS (“Randomized” CLARA)

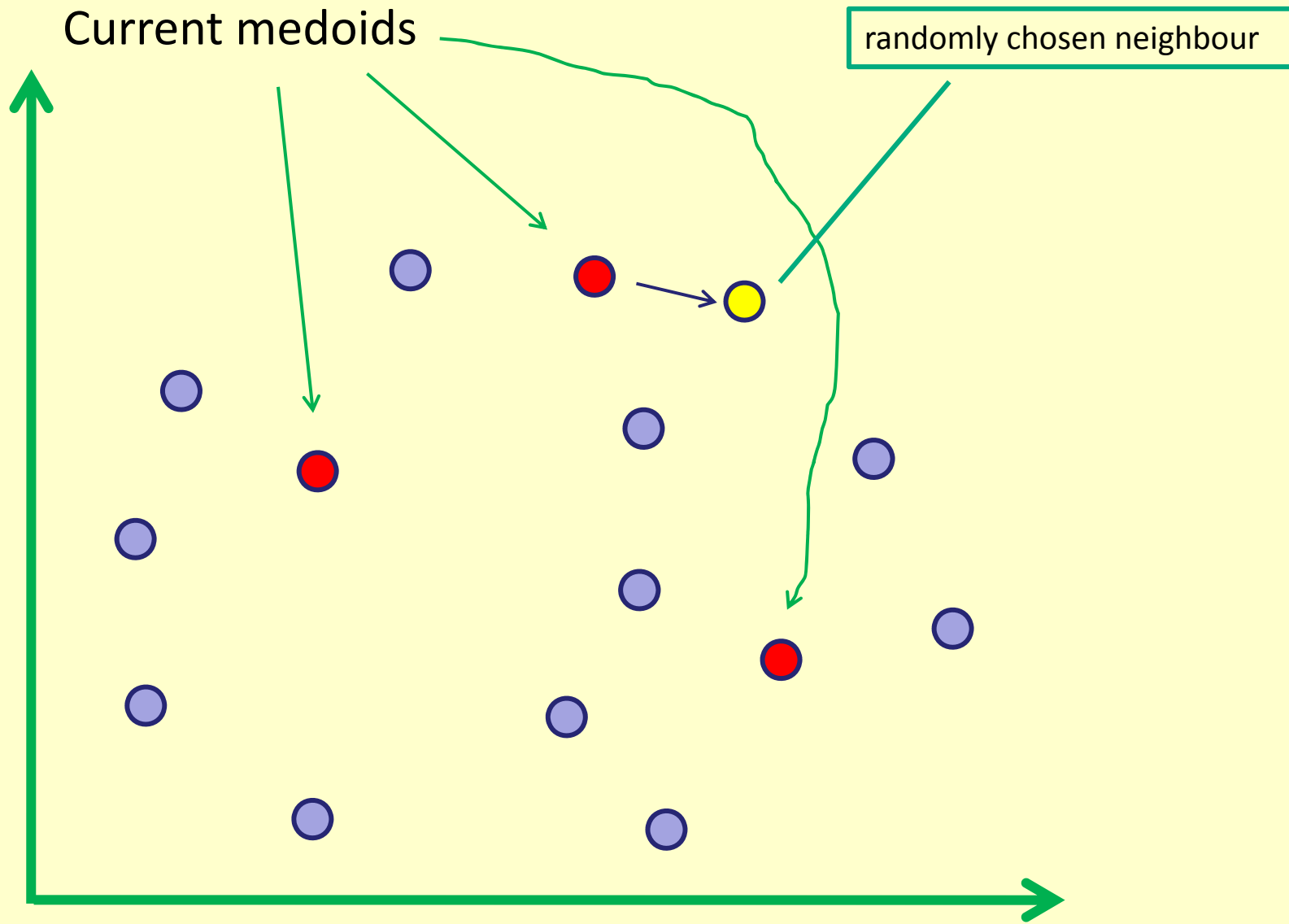
- *CLARANS* (A Clustering Algorithm based on Randomized Search) (Ng and Han’94)
- The clustering process can be presented as **searching a graph** where every **node is a potential solution**, that is, a set of k medoids
- Two nodes are **neighbours** if their sets **differ by only one medoid**
- Each **node** can be **assigned a cost** that is defined to be the **total dissimilarity between every object** and the medoid of its cluster
- The problem corresponds to **search for a minimum on the graph**
- At each step, all neighbours of `current_node` node are searched; the **neighbour which corresponds to the deepest descent in cost** is chosen as the next solution

CLARANS (“Randomized” CLARA)

- For large values of n and k , examining $k(n-k)$ neighbours is time consuming.
- At each step, CLARANS draws sample of neighbours to examine.
- Note that CLARA draws a sample of nodes at the beginning of search; therefore, CLARANS has the benefit of not confining the search to a restricted area.
- If the local optimum is found, CLARANS starts with a new randomly selected node in search for a new local optimum. The number of local optimums to search for is a parameter.
- It is more efficient and scalable than both PAM and CLARA; returns higher quality clusters.

Algorithm CLARANS

- 1. Input parameters *num_local_min* and *max_neighbor*. Initialize *i* to 1, and *mincost* to a large number.
- 2. Set *current_node* to an arbitrary node in $G_{n;k}$.
- 3. Set *j* to 1.
- 4. Consider a random neighbor *S* of *current_node*, and based on *S*, calculate the cost differential of the two nodes.
- 5. If *S* has a lower cost, set *current_node* to *S*, and go to Step 3.
- 6. Otherwise, increment *j* by 1. If $j \leq \text{max_neighbor}$, go to Step 4.
- 7. Otherwise, when $j > \text{max_neighbor}$, compare the cost of *current_node* with *mincost*. If the former is less than *mincost*, set *mincost* to the cost of *current_node* and set *bestnode* to *current_node*.
- 8. Increment *i* by 1. If $i > \text{num_local_min}$, output *bestnode* and halt. Otherwise, go to Step 2.



A node is a set of k objects. A neighbor of a node is a node that differs by just one element.

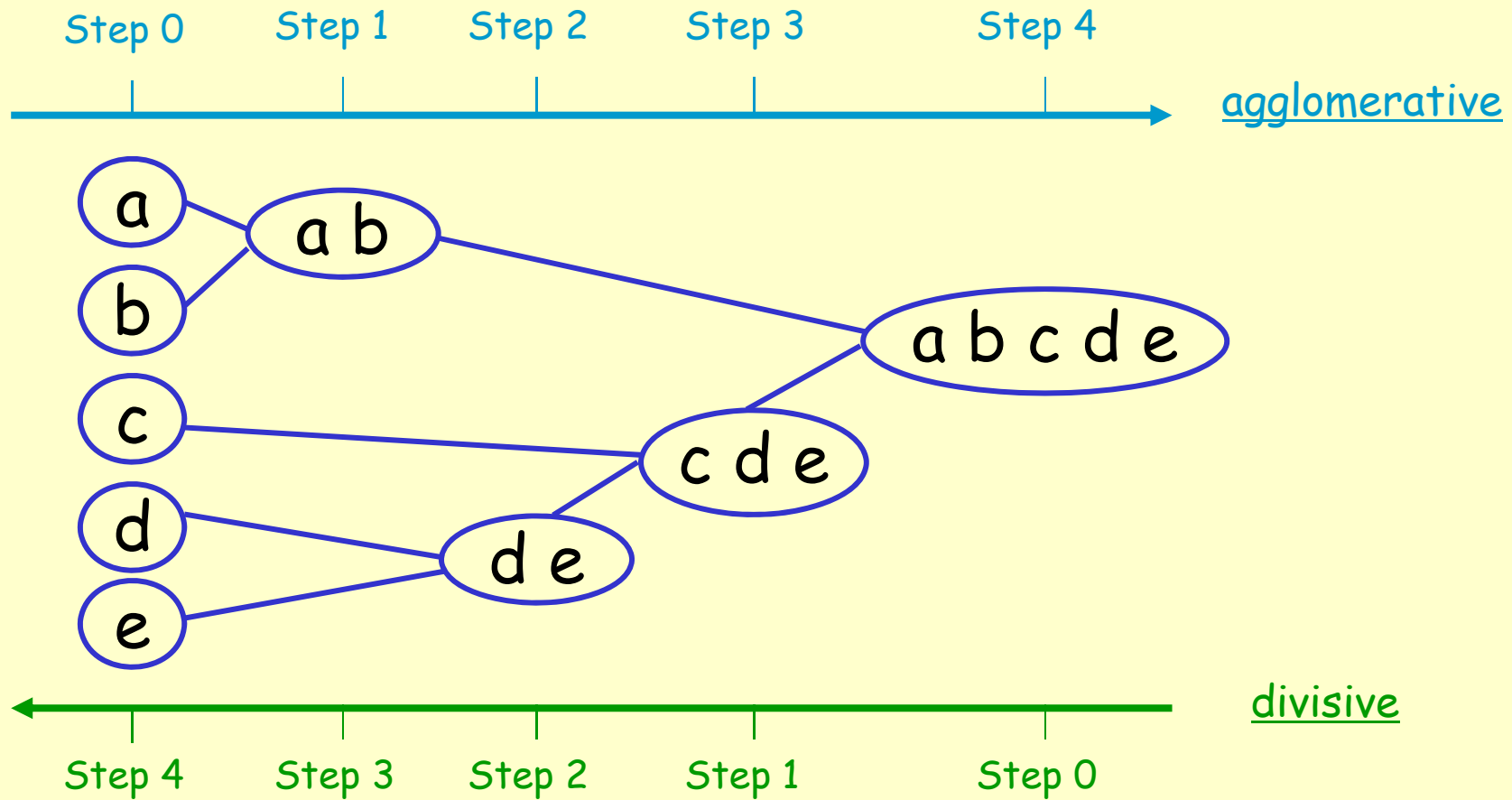
Clustering

- What is Cluster Analysis?
- Types of Data in Cluster Analysis
- A Categorization of Major Clustering Methods
- Partitioning Methods
- Hierarchical Methods

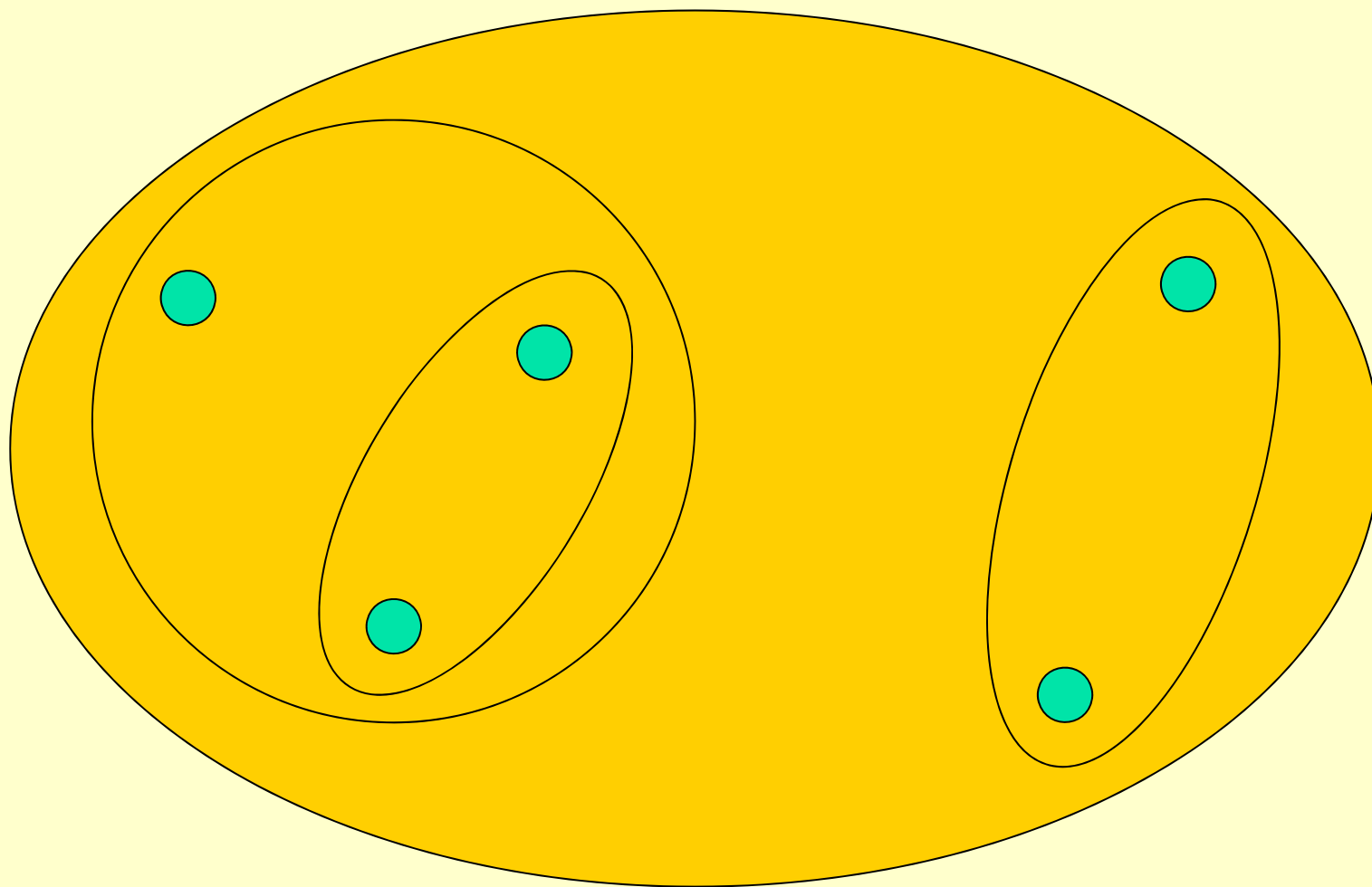
Hierarchical Clustering

- Use distance matrix as clustering criteria.
- These methods work by grouping data into a tree of clusters.
- There are two types of hierarchical clustering:
 - **Agglomerative**: bottom-up strategy
 - **Divisive**: top-down strategy
- Does not require the number of clusters as an input, but **needs a termination condition**, e.g., could be the desired number of clusters or a distance threshold for merging

Hierarchical Clustering













Agglomerative hierarchical clustering



We begin with a distance matrix which contains the distances between every pair of objects in our database.

$$D(\text{Mrs. Muntz}, \text{Lisa Simpson}) = 8$$

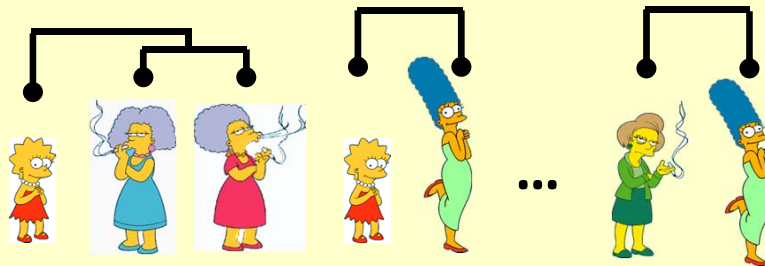
$$D(\text{Auntie Platter}, \text{Auntie Platter}) = 1$$

				
	0	8	7	7
		0	4	4
			0	3
				0
				

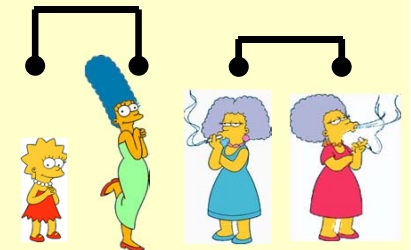
Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

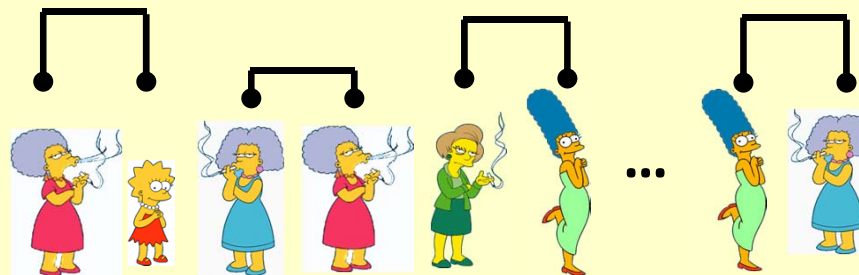
Consider all possible merges...



Choose the best



Consider all possible merges...



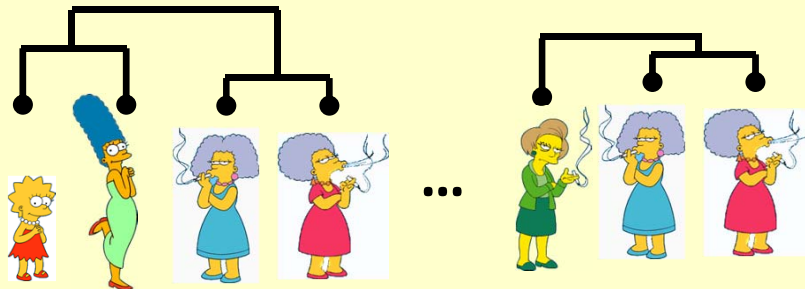
Choose the best



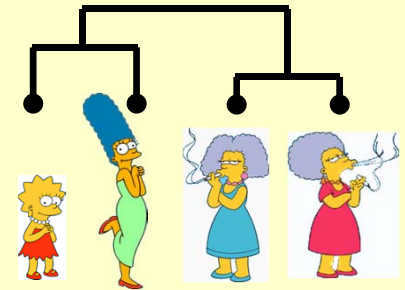
Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

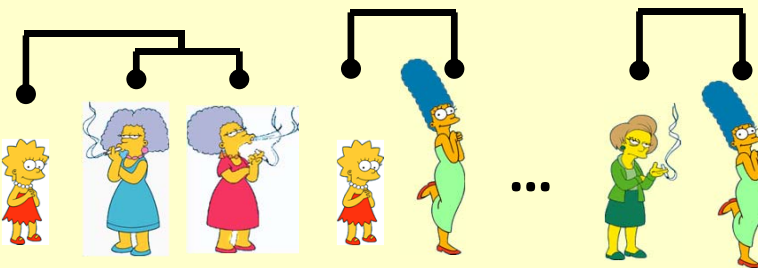
Consider all possible merges...



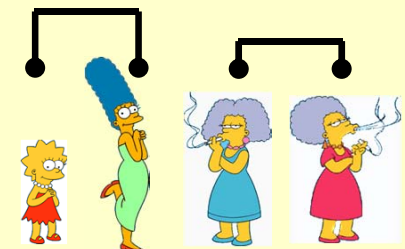
Choose the best



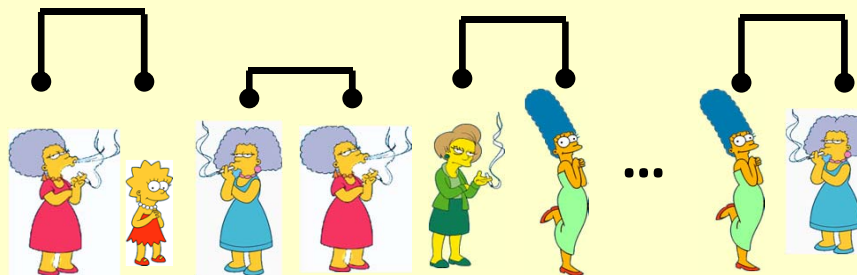
Consider all possible merges...



Choose the best



Consider all possible merges...



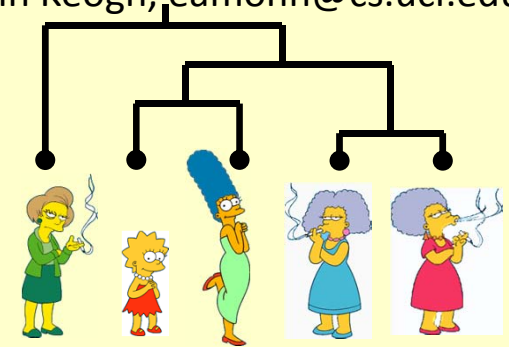
Choose the best



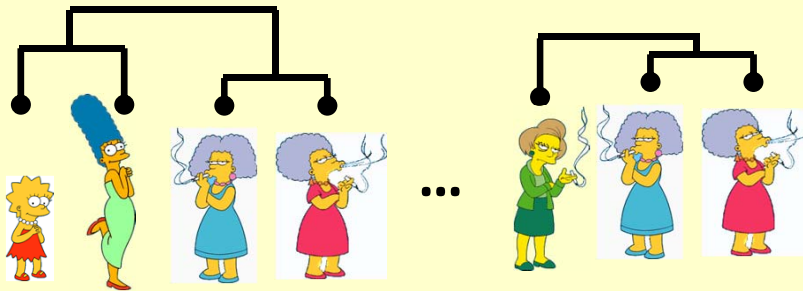
Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

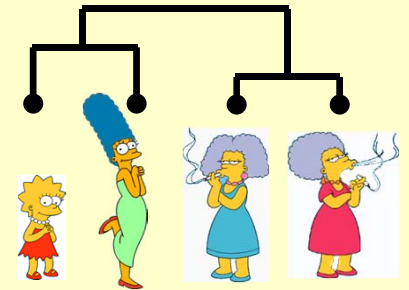
(c) Eamonn Keogh, eamonn@cs.ucr.edu



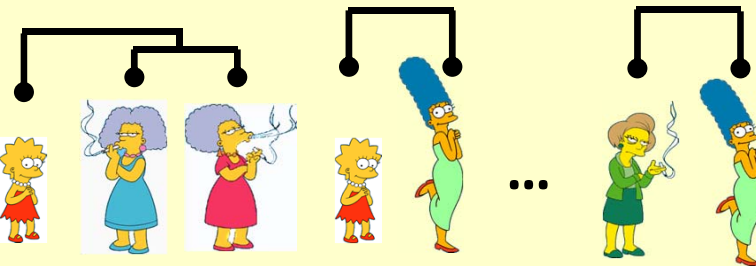
Consider all possible merges...



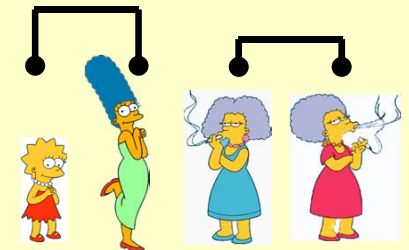
Choose the best



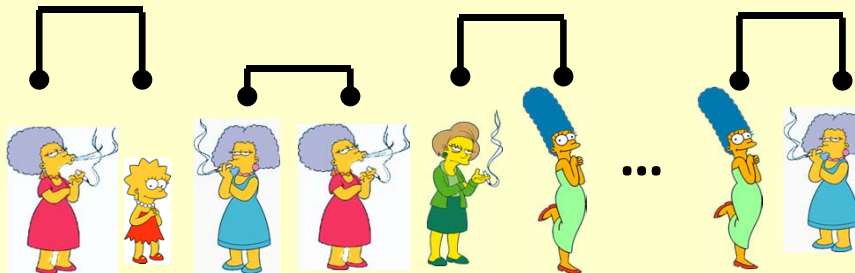
Consider all possible merges...



Choose the best



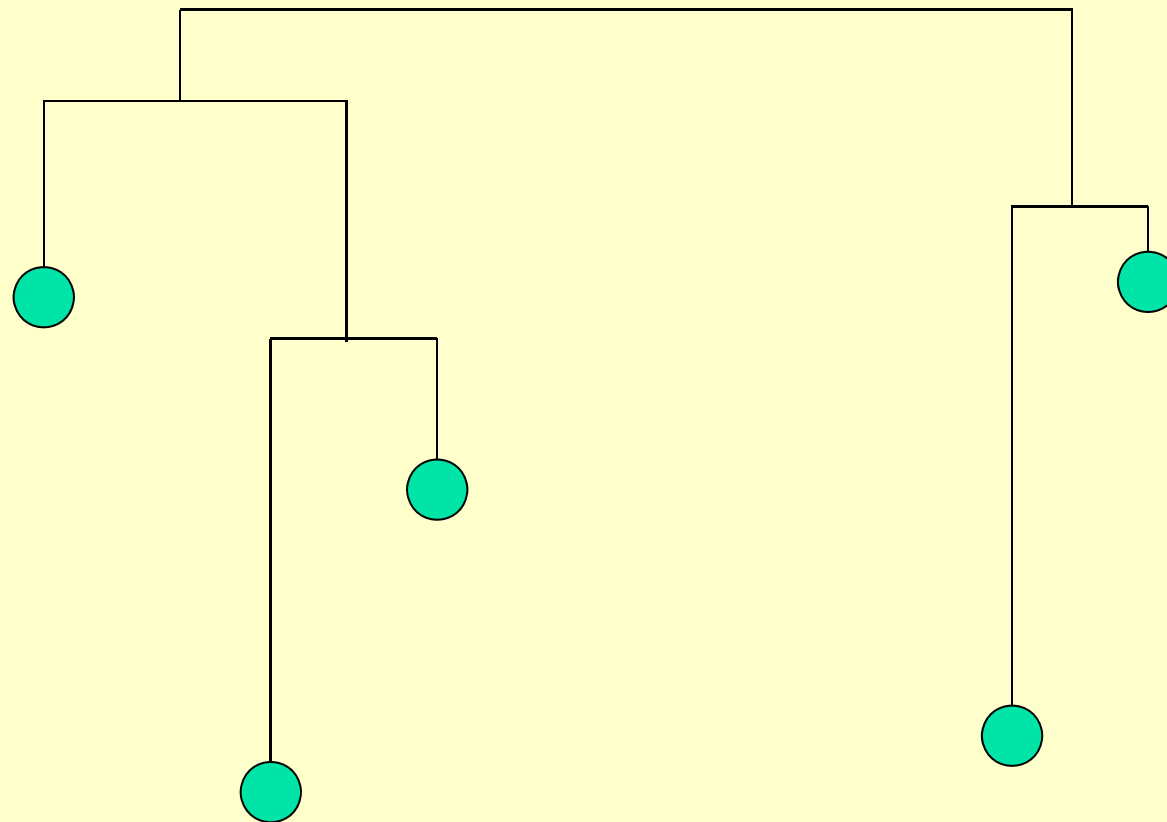
Consider all possible merges...



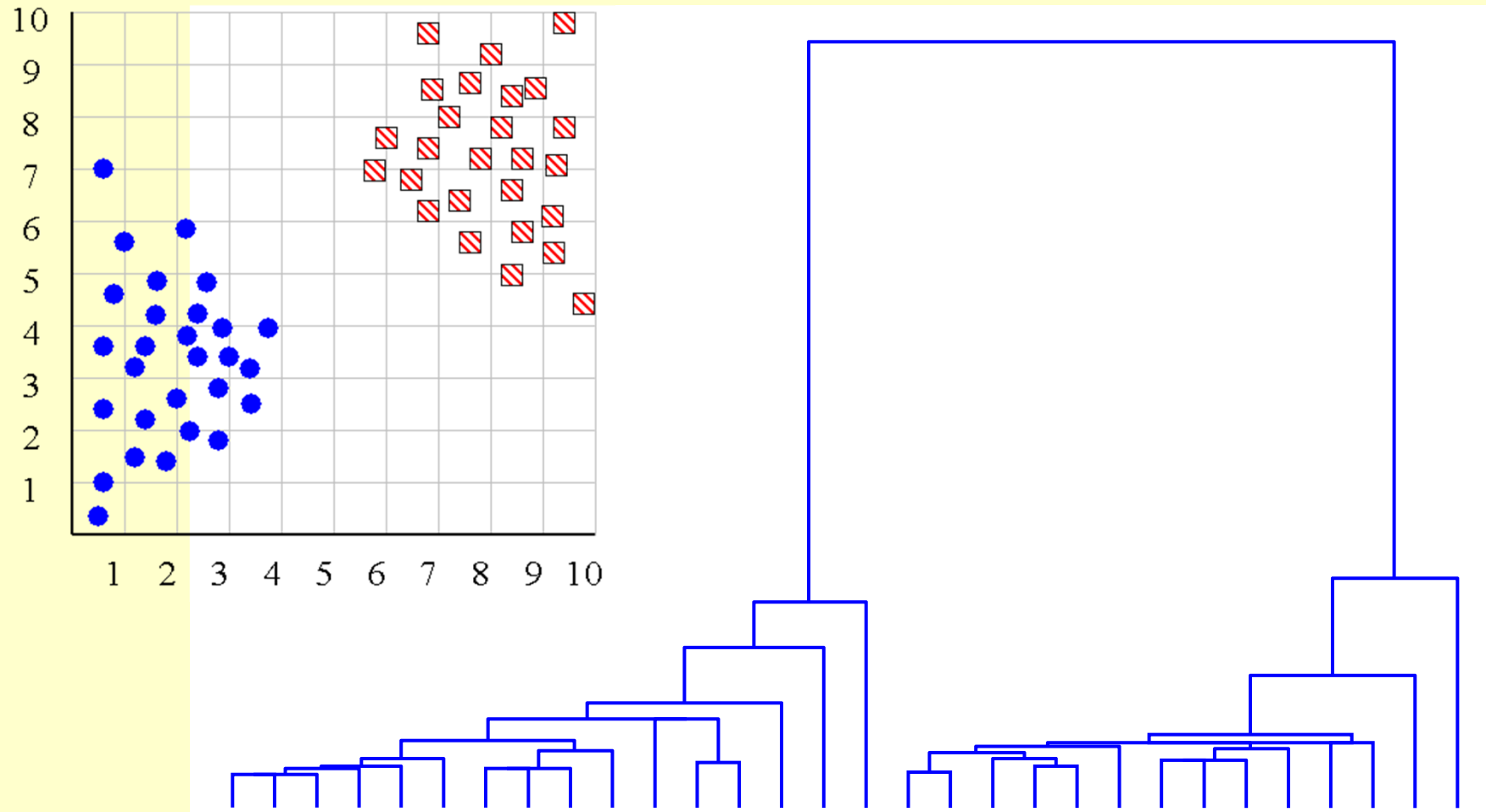
Choose the best



Clustering result: dendrogram

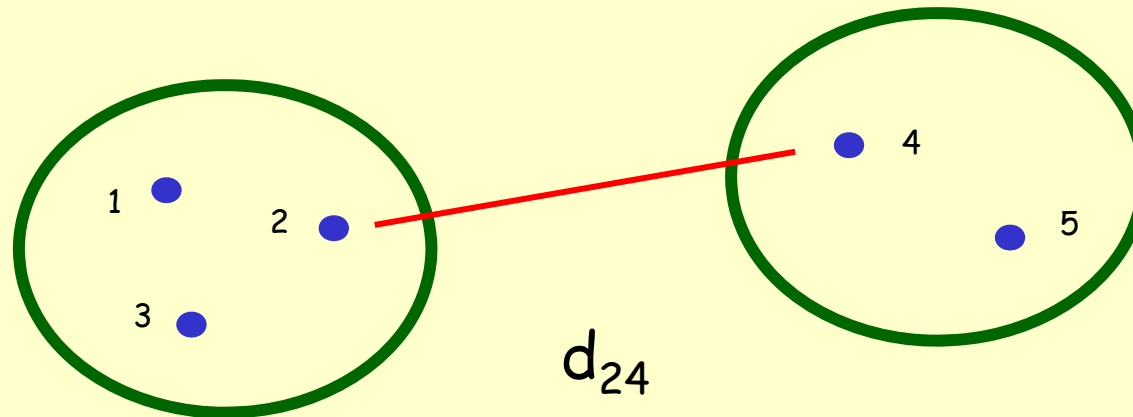


We can look at the dendrogram to determine the “correct” number of clusters. In this case, the two highly separated subtrees are highly suggestive of two clusters. (Things are rarely this clear cut, unfortunately)



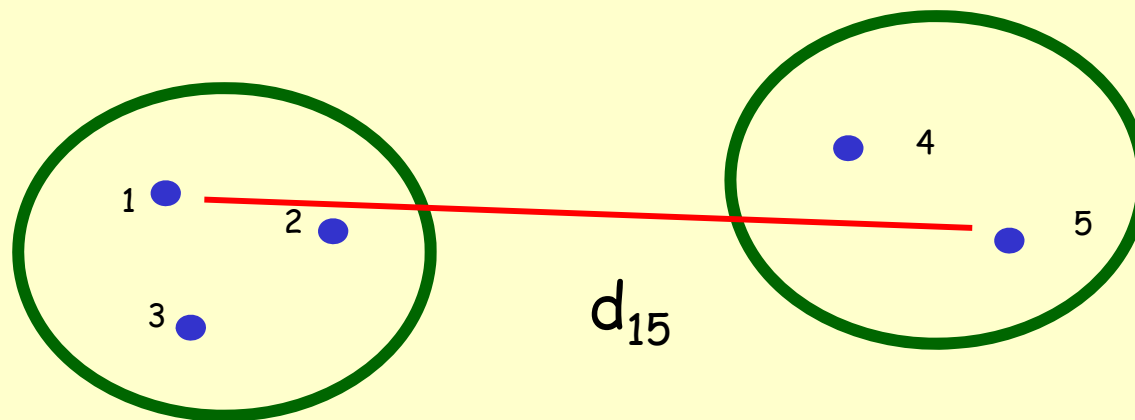
Linkage rules (1)

- **Single link (nearest neighbour)**. The distance between two clusters is determined by the distance of the two closest objects (nearest neighbours) in the different clusters.
 - This rule will, in a sense, *string* objects together to form clusters, and the resulting clusters tend to represent long "chains."



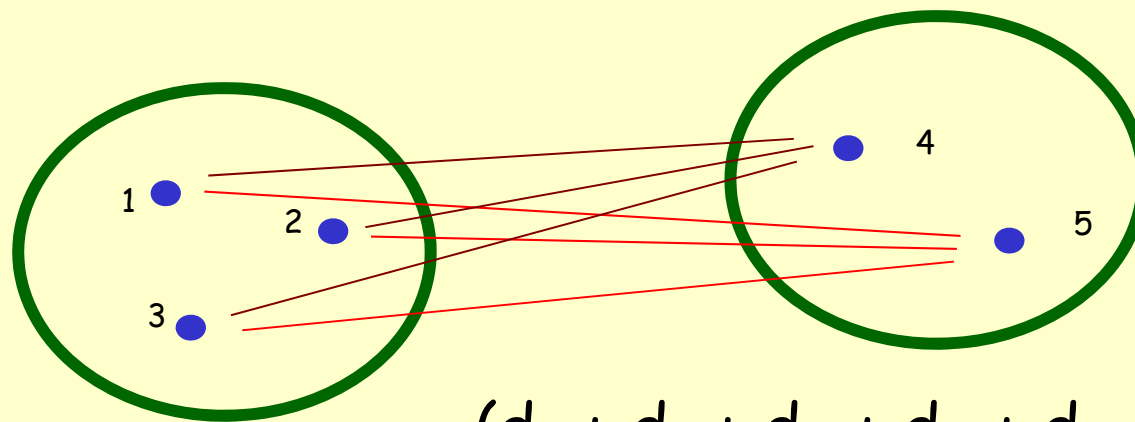
Linkage rules (2)

- **Complete link (furthest neighbour).** The distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbours").
 - This method usually performs quite well in cases when the objects actually form naturally distinct "clumps." If the clusters tend to be somehow elongated or of a "chain" type nature, then this method is inappropriate.



Linkage rules (3)

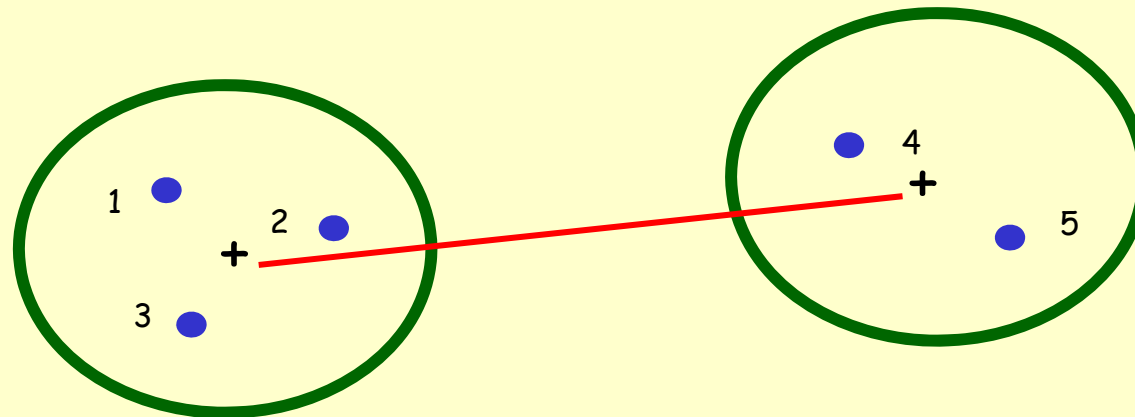
- **Pair-group average.** The distance between two clusters is calculated as the **average distance between all pairs of objects** in the two different clusters. This method is also very efficient when the objects form natural distinct "clumps," however, it performs equally well with elongated, "chain" type clusters.



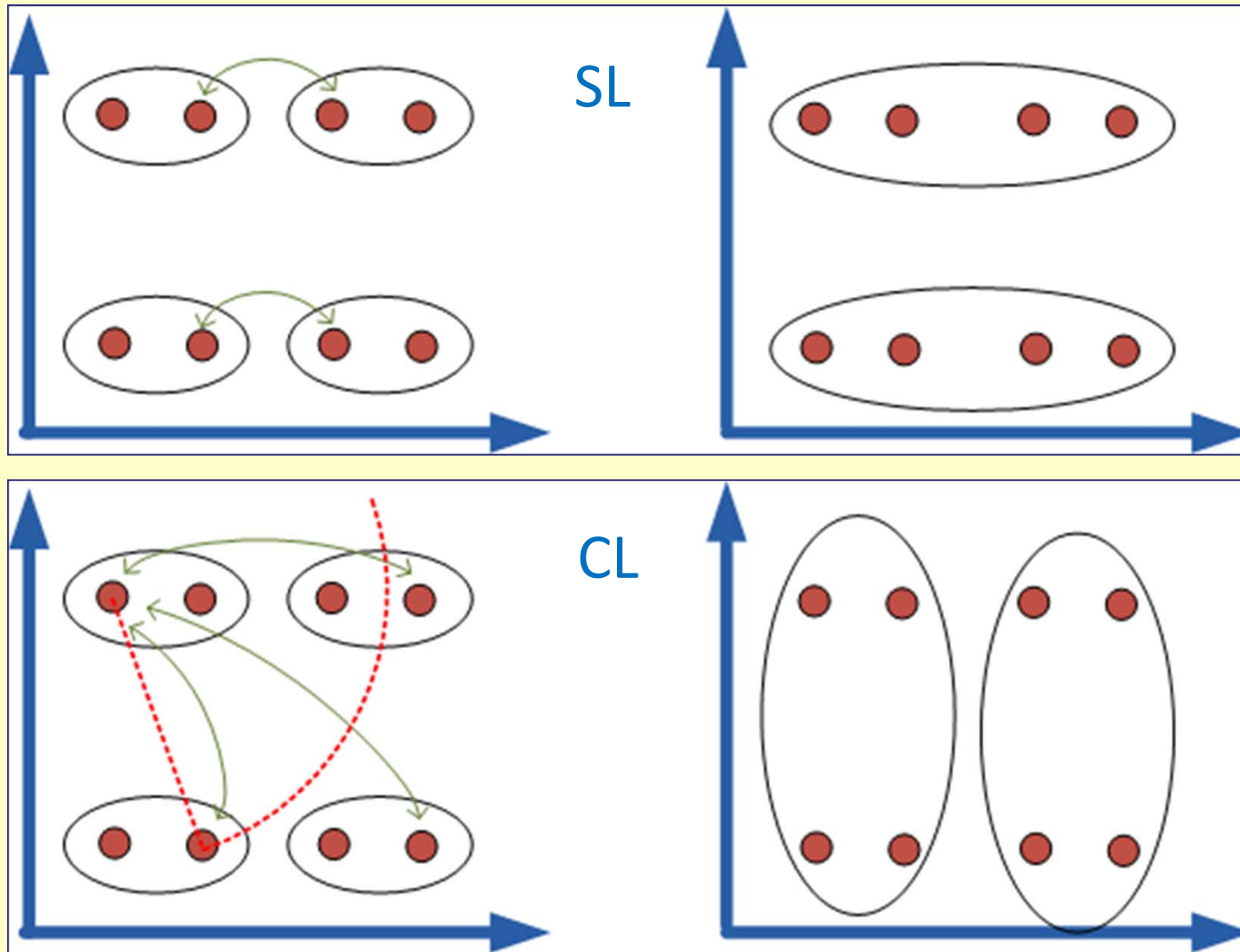
$$(d_{14} + d_{15} + d_{24} + d_{25} + d_{34} + d_{35}) / 6$$

Linkage rules (4)

- **Pair-group centroid.** The distance between two clusters is determined as the **distance between centroids**.

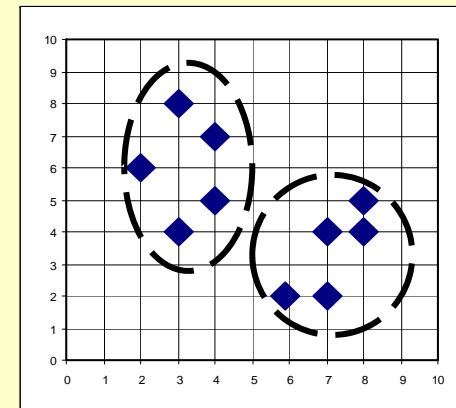
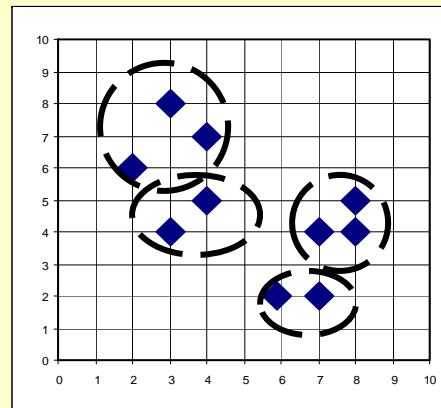
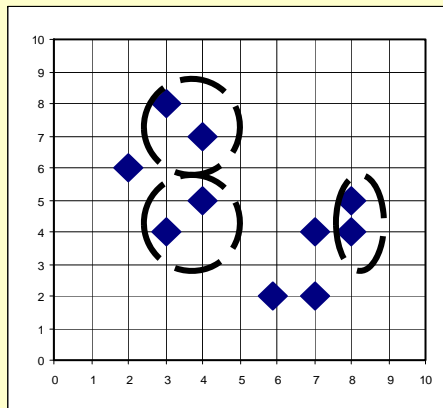


Comparing single and complete link



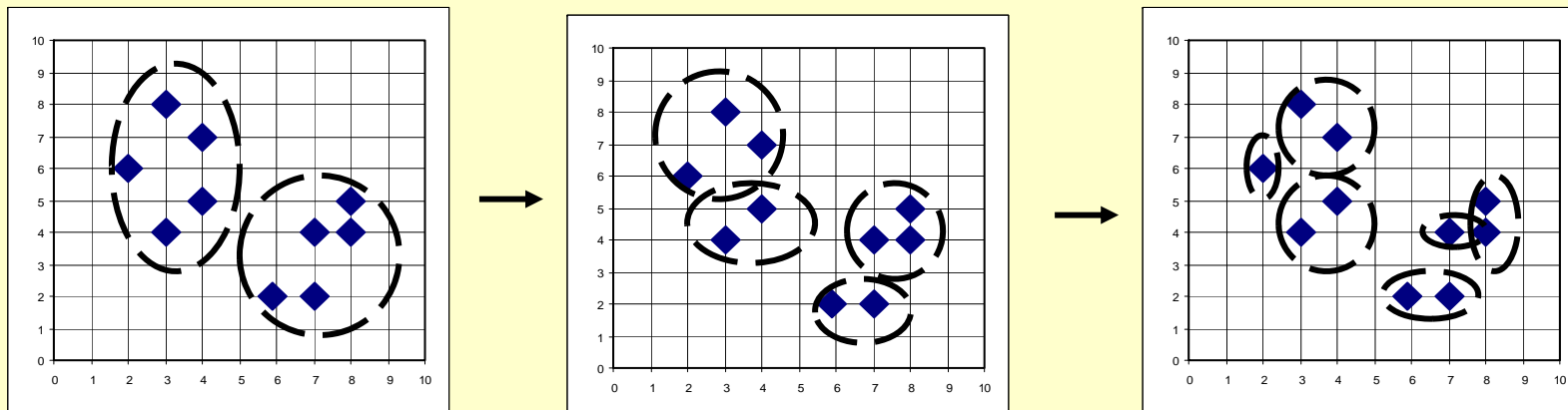
AGNES (Agglomerative Nesting)

- Use the **Single-Link** method and the dissimilarity matrix.
- Repeatedly **merge nodes that have the least dissimilarity**
 - merge C1 and C2 if objects from C1 and C2 give the minimum Euclidean distance between any two objects from different clusters.
- Eventually all nodes belong to the same cluster



DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Inverse order of AGNES
 - All objects are used to form **one initial cluster**
 - The largest cluster is **split according to some principle**
 - the maximum Euclidean distance between the closest neighbouring objects in different clusters
- Eventually each node forms a cluster on its own



More on Hierarchical Clustering

- Do not scale well
 - time complexity of at least $O(n^2)$, where n is the number of total objects
- Can never undo what was done previously
- It's nice that you get a hierarchy instead of an amorphous collection of groups
- Don't need to specify k
 - If you want k groups, just cut the $(k-1)$ longest links
- In general give better quality clusters than k -means' like methods

More on Hierarchical Clustering

- Integration of hierarchical with distance-based clustering
 - **BIRCH**: uses CF-tree and incrementally adjusts the quality of sub-clusters
 - **DBscan**: **Density-based Alg.** based on local connectivity and density functions

BIRCH algorithm

- **BIRCH**: Balanced Iterative Reducing and Clustering using Hierarchies, by Zhang, Ramakrishnan, Livny (SIGMOD'96)
 - A tree is built that captures needed information to perform clustering
 - Introduces two new concepts
 - **Clustering Feature** (*contains info about a cluster*)
 - **Clustering Feature Tree**
- which are used to summarize cluster representation

BIRCH - Clustering Feature Vector

A **clustering feature** is a triplet summarizing information about sub-clusters of objects.

It registers **crucial measurements** for computing **clusters in a compact form**

Clustering Feature: $CF = (N, LS, SS)$

N : Number of data points

LS Linear sum: $\sum_{i=1}^N X_i$

SS Square Sum: $\sum_{i=1}^N X_i^2$

Clustering Feature Vector - example

(3,4)

(2,6)

(4,5)

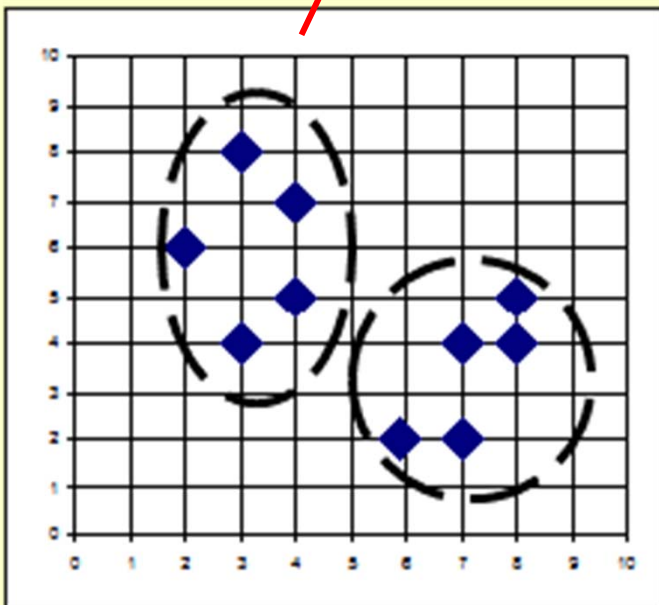
(4,7)

(3,8)

$$N = 5$$

$$LS = (3+2+4+4+3, \\ 4+6+5+7+8) = (16, 30)$$

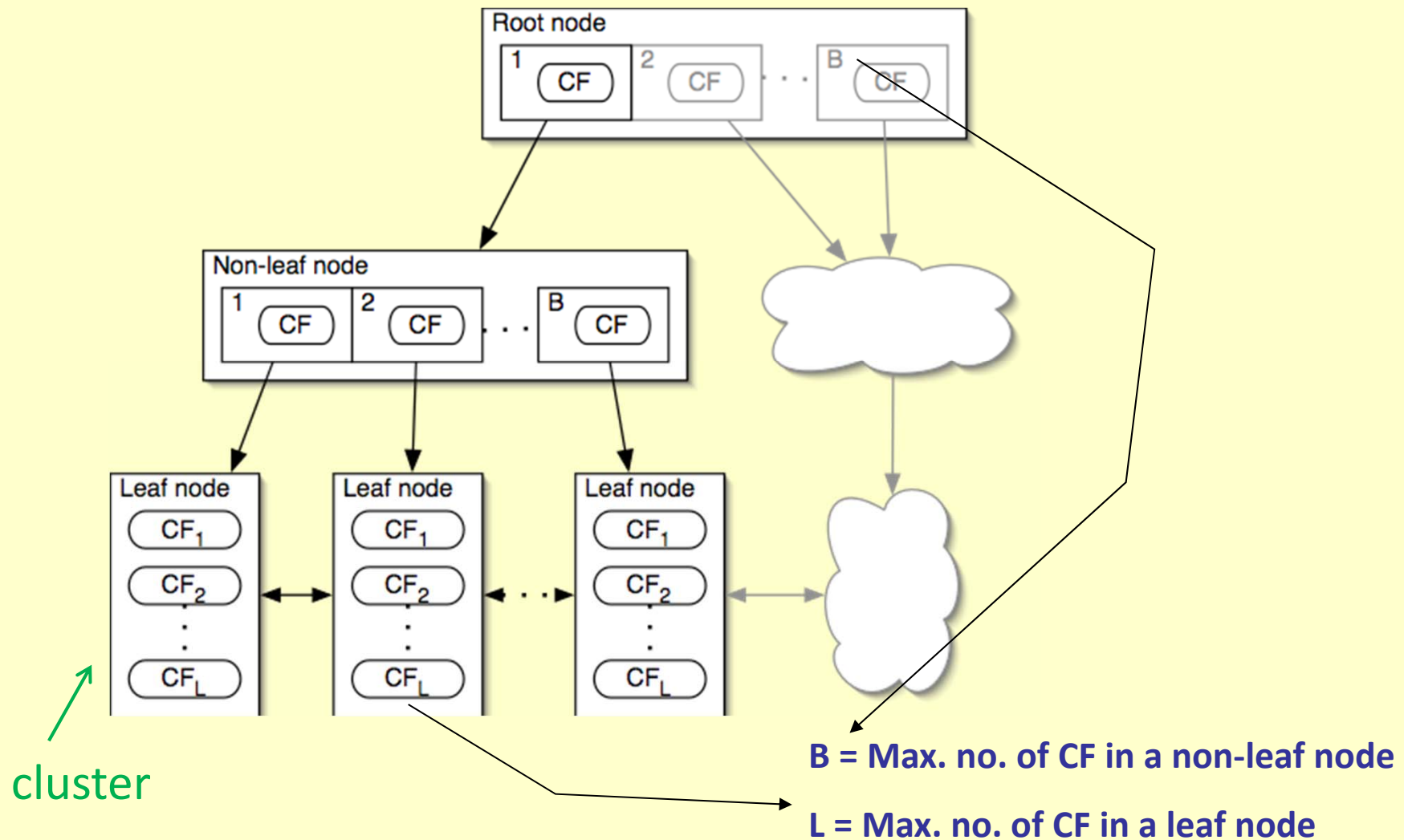
$$SS = (3^2+2^2+4^2+4^2+3^2, \\ 4^2+6^2+5^2+7^2+8^2) \\ = (54, 190)$$



$$CF = (5, (16, 30), (54, 190))$$

BIRCH - Clustering Feature Tree

A **tree** that stores the **clustering features** for hierarchical clustering



Notes on Birch

- A Leaf node represents a cluster.
- A sub-cluster in a leaf node must have a **diameter no greater** than a given **threshold T**.
- A **point is inserted** into the **leaf node** (cluster) to which is **closer**.
- When one item is inserted into a cluster at the leaf node, the restriction T (for the corresponding sub-cluster) must be satisfied. The corresponding CF must be updated.
- **If there is no space** on the node **the node is split**.

BIRCH algorithm

- **Incrementally construct a CF tree**, a hierarchical data structure for multiphase clustering
- **Phase 1**: scan DB to build an initial in-memory CF tree
 - If threshold condition is violated
 - If there is room to insert – Insert point as a single cluster
 - If not
 - Leaf node split: take two farthest CFs and create two leaf nodes, put the remaining CFs (including the new one) into the closest node
 - Update CF for non-leaves. Insert new non-leaf entry into parent node
 - We may have to split the parent as well. Split the root increases tree height by one.
 - If not
 - Insert point into the closest cluster
- **Phase 2**: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree

Some Comments on Birch

- It can be shown that CF vectors can be stored and calculated incrementally and accurately as clusters are merged
- Experiments have shown that scales linearly with the number of objects.
- Finds a good clustering with a single scan and improves the quality with a few additional scans
- Handles only numeric data, and sensitive to the order of the data record.
- Better suited to find spherical clusters.

DBSCAN algorithm

- **Density-based Alg:** based on local connectivity and density functions
- Major features:
 - Discover clusters of arbitrary shape
 - Handle noise
 - One scan

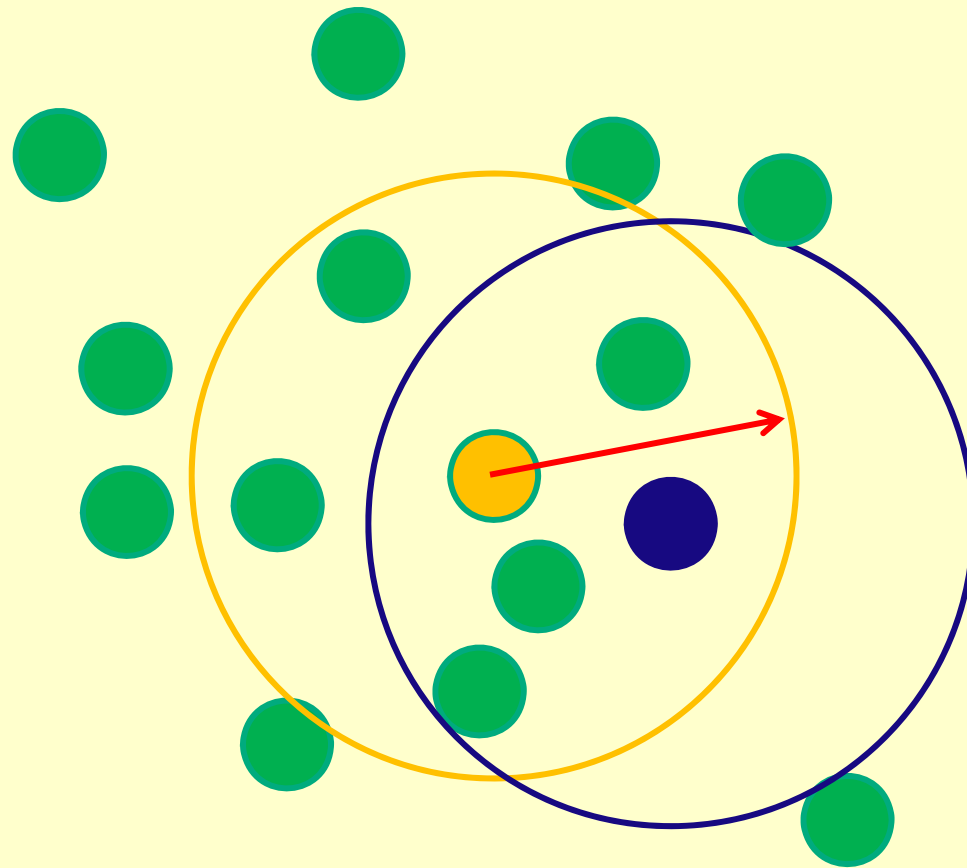
DBSCAN: Density-Based Clustering





- Clustering based on density (local cluster criterion), such as density-connected points
- Each cluster has a considerable higher density of points than outside of the cluster

DBSCAN: Density Concepts (1)

- **Density**: the minimum number of points within a certain distance of each other.
- Two parameters:
 - **Eps** : Maximum radius of the neighbourhood
 - **MinPts** : Minimum number of points in an Eps-neighbourhood of that point
- **Core Point**: object with at least **MinPts** objects within a **radius** 'Eps-neighbourhood'



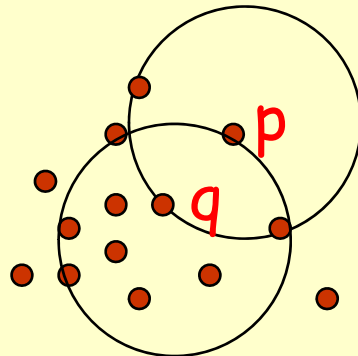
 Core point for $EPS=4cm$ and $MinPts = 5$

 Not a Core point

DBSCAN: Density Concepts (2)

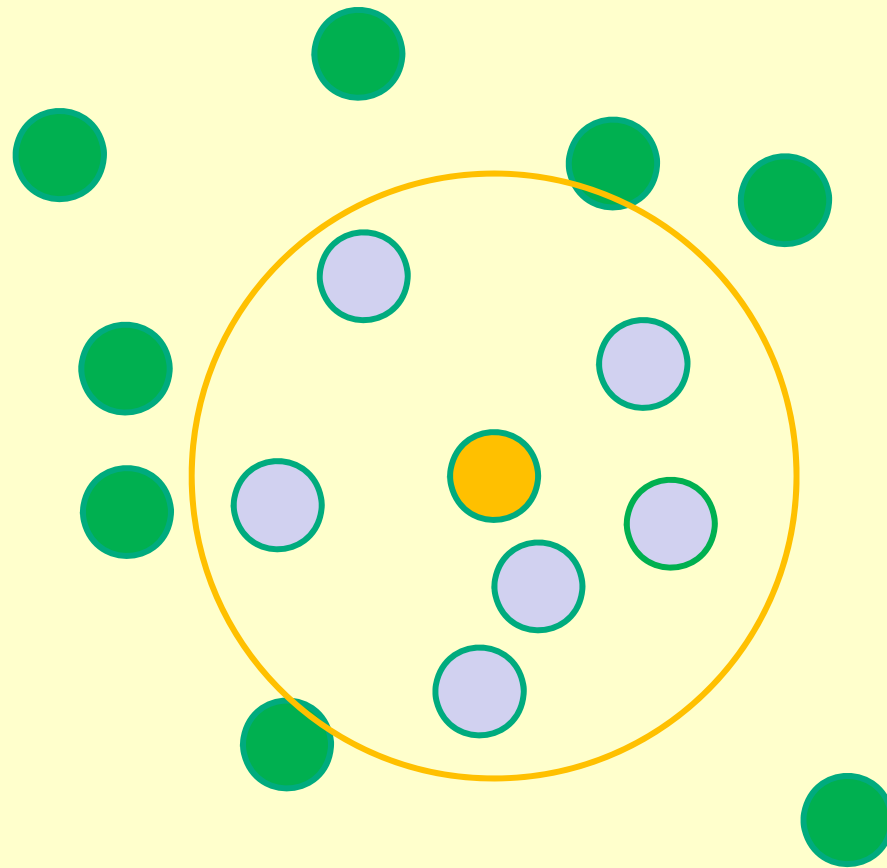
- **Directly Density-Reachable**: A point p is directly density-reachable from a point q with respect to Eps , $MinPts$ if
 - 1) p belongs to $NEps(q)$
 - 2) core point condition: $|NEps(q)| \geq MinPts$

*(a DDR point needs to be close to a core point but it does not need to be a core point itself, if not it is a **border point**)*



MinPts = 5

Eps = 1 cm



Core point

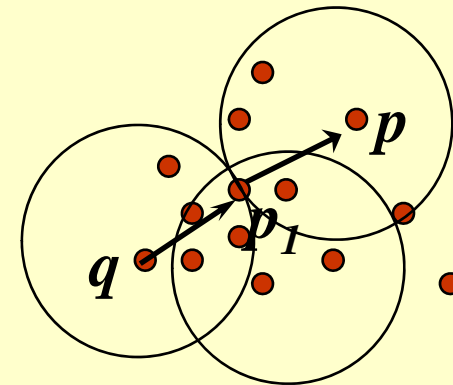


Points that are directly density reachable from the core point

DBSCAN: Density Concepts (2)

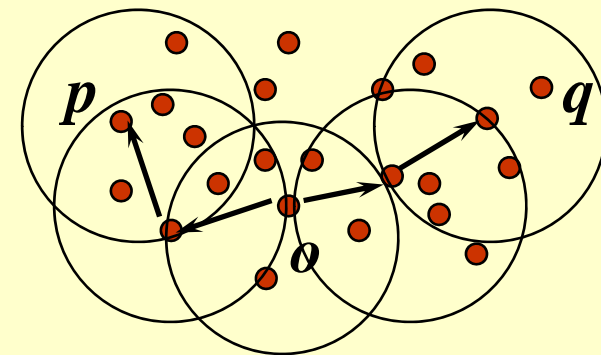
- **Density-reachable:**

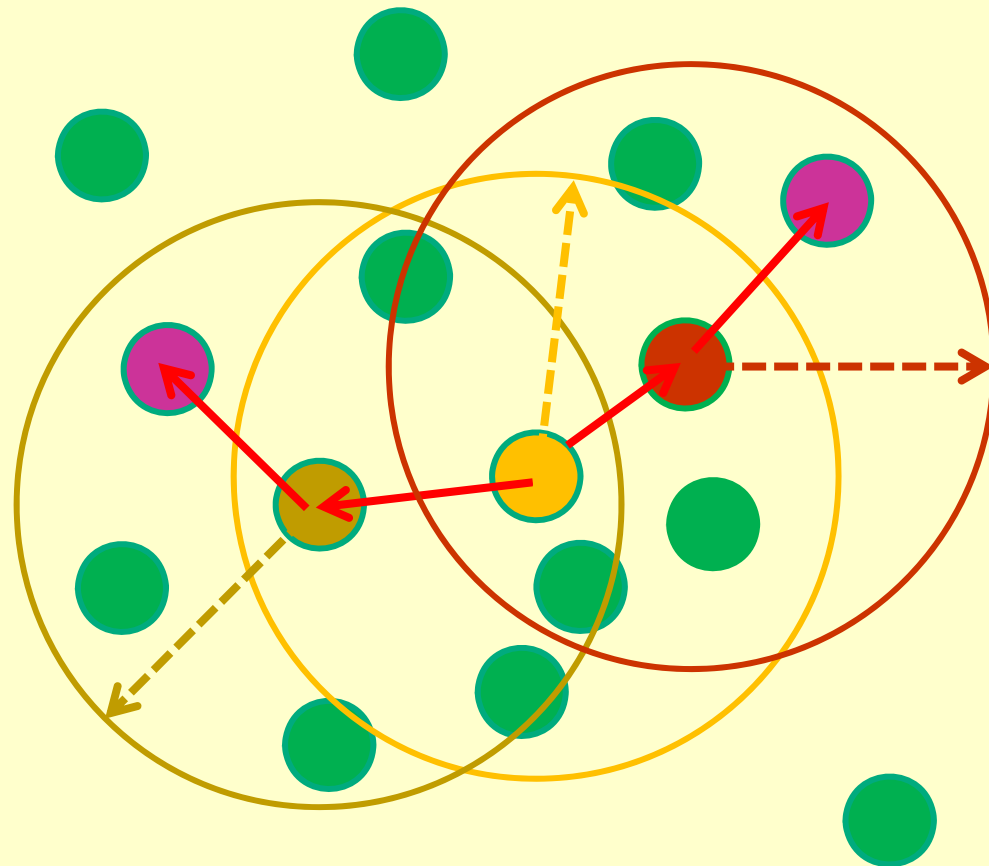
- A point p is **density-reachable** from a point q wrt. Eps , MinPts if there is a chain of points p_1, \dots, p_n , $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i



- **Density-connected:**

- A point p is **density-connected** to a point q wrt. Eps , MinPts if there is a point o such that both, p and q are density-reachable from o wrt. Eps and MinPts .





- Core points
- Density reachable from
- Density connected to

DBSCAN: Cluster definition

- A *cluster* is defined as a **maximal set of density-connected points**
- A cluster has a core set of points very close to a large number of other points (core points) and then some other points (border points) that are sufficiently close to at least one core point.

Intuition on DBSCAN

- For each **core point** which is not in a cluster
 - Explore its neighbourhood in search for every density reachable point
 - For **each neighbourhood** point explored
 - If it is a **core point** -> **further explore** it
 - If it is **not a core point** -> **assign to the cluster** and do not explore it
- The fact that a cluster is composed by the maximal set of points that are density-connect it is a property (and therefore a consequence) of the method

DBSCAN: The Algorithm

- Arbitrary **select** a point p
- If p is not a core point, no points are density-reachable from p and DBSCAN visits the next point of the database.
- If p is a core point, a cluster is formed.
 - **Retrieve** all points **density-reachable from p** wrt **Eps** and **$MinPts$** .
- Continue the process until all of the points have been processed.

(it is possible that a border point could belong to two clusters. Such point will be assigned to whichever cluster is generated first)

DBScan

- Experiments have shown DBScan to be faster and more precise than CLARANS
- Expected time complexity $O(n \lg n)$

Clustering Summary

- Unsupervised method to find groups of instances
- Many approaches
 - partitioning
 - hierarchical
- Solution evaluation is difficult
 - Manual inspection by experts
 - Benchmarking on existing labels
 - Cluster quality measures
 - (measure the "tightness" or "purity" of clusters)
 - distance measures
 - high similarity within a cluster, low across clusters

References

- **Jiawei Han and Micheline Kamber**, “Data Mining: Concepts and Techniques”, 2 edition (4 Jun 2006),
- **Gordon S. Linoff and Michael J. Berry**, “Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management”, 3rd Edition edition (1 April 2011)