Clustering 7.4 – 7.6

- **7.4 Partitioning methods**
- **7.5 Hierarchical methods**
- **7.6 Density-based methods**

1

7.4 Partitioning Methods

- Given a dataset, *D,* and k, the number of clusters to form, a partitioning algorithm organizes *n* objects into *k* partitions, where (k <= n)
- Each partition represents a cluster
- These clusters are formed to optimize an objective partitioning criterion, in terms of attributes
- dissimilarity function (based on distance), where “similar” objects reside within a cluster and “dissimilar” objects reside in other clusters

2

k-Means and *k*-Medoids

- Heuristic methods: *k*-means and *k*-medoids
  - *k*-means: a cluster is represented by the mean (center) value
  - *k*-medoids or PAM (partition around medoids): each cluster is represented by one of the objects in the cluster
- Centroid-Based Technique: *k*-Means Method

3

k-Means and *k*-Medoids

- Partitions a set of n objects into k clusters,
  - optimizing the *intra*cluster similarity
  - minimizing *inter*cluster similarity
- Cluster similarity is measured with respect to the mean value of the objects within that cluster → referred to as the cluster's *centroid* or *center of gravity*

4

k-Means algorithm

- Inputs → *D* – dataset with *n* objects
- Output → set of *k* clusters
- Method
  - (1) arbitrarily select *k* objects from D as the initial cluster mean (centroid)
  - (2) repeat
  - (3) reassign each object to the cluster to which it is most similar, based upon the distance between the object and the cluster mean
  - (4) update cluster mean for objects in each cluster
  - (5) iterate until the criterion function converges (no redistribution of objects in any cluster)
    – known as *iterative relocation*

5

k-Means algorithm

- Use the squared-error criterion – distance from object to centroid is squared and then distances are summed

(below is the absolute-error criterion used in k-Medoids)

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} d(p, o_i)^2$$

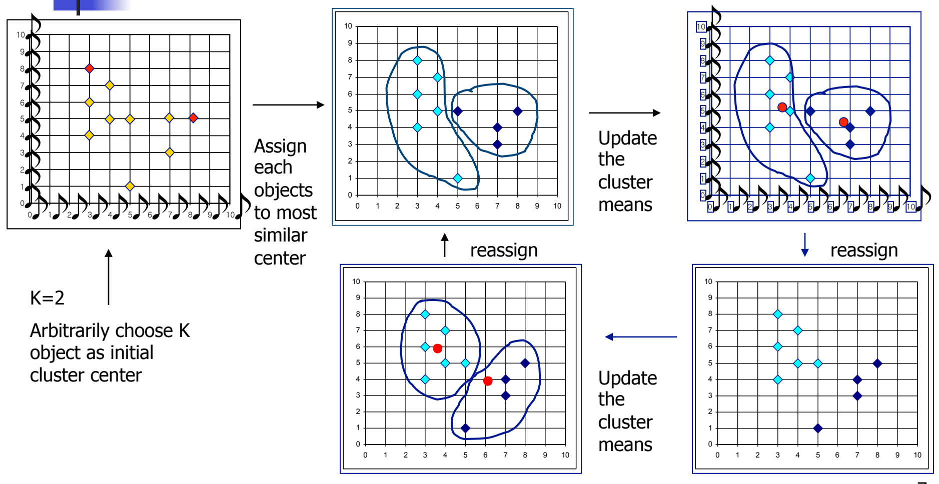
→ E = sum of square error for all objects

→ p = point in space representing object

→ m_i = mean of cluster, C_i

6

k-Means: Example


Assign each objects to most similar center
Update the cluster means
reassign
reassign
Update the cluster means
K=2
Arbitrarily choose K object as initial cluster center

Pros and Cons of *k*-Means

- Relatively efficient – computational complexity: O(tkn)
  - n = # objects
  - k = # clusters
  - T = # iterations
  - k, t << n.

Pros and Cons of *k*-Means

- Applicable only when mean is defined
  - What about categorical data? – NO (*k*-modes)
  - Can we normalize occurrence of categorical data? – We might lose some attribute details??
  - Create binary values for categorical data? – We might lose some attribute details??
- Need to specify the number of clusters
- Unable to handle noisy data and outliers

9

Variations of the *k*-Means

- Aspects
  - Selection of the initial *k*-means
  - Calculation of dissimilarity
  - Different strategies to calculate cluster means:
  - → one example is to apply a hierarchical agglomerative algorithm, which determines the number of clusters and finds an initial clustering, and then use iterative relocation to improve clustering

10

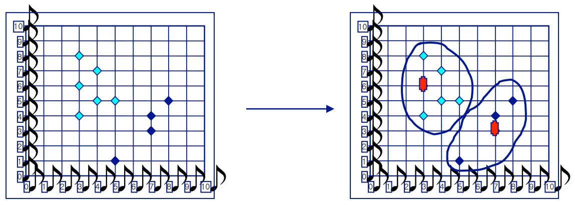
Variations of the *k*-Means

- *k*-modes handle categorical data
  - Use mode instead of mean
    - Mode: the most frequently occurring item
  - A mixture of categorical and numerical data: k-prototype method
- Expectation-Maximum
  - Each object is assigned to each cluster, according to a weight representing the probability of membership in that particular cluster
  - Details → 7.8 Model-Based Clustering Methods

11

Representative Object-Based Technique: *k*-medoids method

- *k*-medoids: the most centrally located object in a cluster
- Used because *k*-means is sensitive to outliers – large values distort the distribution of objects



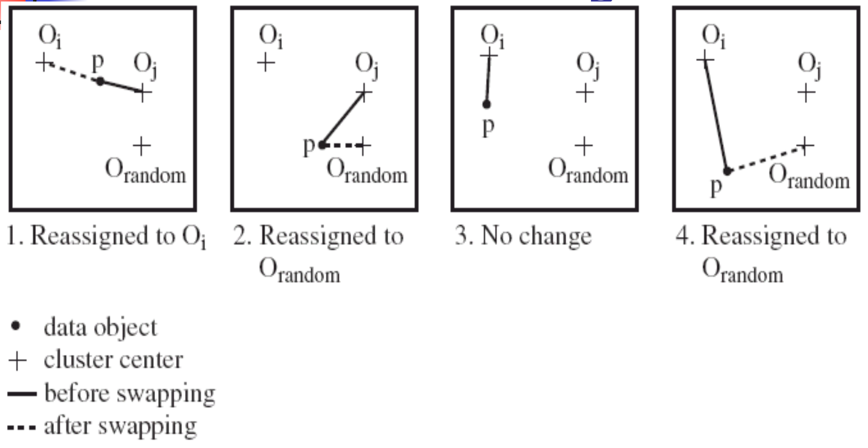
12

Partitioning Around Medoids (PAM)

- Instead of mean value of objects in the cluster, we arbitrarily choose an object itself, as the initial medoid
- Until no change, do
  - Reassign each object to the cluster to which the nearest medoid *improves* the clustering
  - For each pair of non-select object O_{random} and selected object O_j, calculate the total swapping cost *S*.
  - If $S < 0$ then swap O_j with O_{random} to form the new set of *k*-medoids.

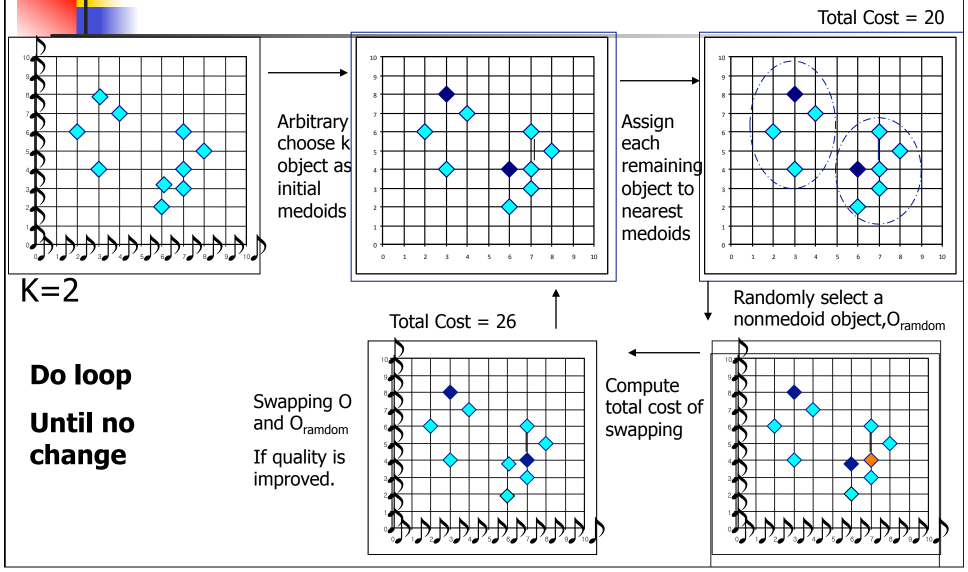
13

k-medoids clustering



14

PAM example


Total Cost = 20
Arbitrary choose k object as initial medoids
Assign each remaining object to nearest medoids
K=2
Randomly select a nonmedoid object,O_{ramdom}
Total Cost = 26
Compute total cost of swapping
Swapping O and O_{ramdom}
If quality is improved.
Do loop
Until no change

Pros and Cons of PAM

- PAM is more robust than k-means in the presence of noise and outliers
  - Medoids are less influenced by outliers
- However, processing *k*-Medoids is more costly (in terms of time and memory)
- Both require the user to specify the number of *k* clusters

Pros and Cons of PAM

- We can also use the k-median (middle value) and k-modes (most frequent value)
- PAM is efficient for small data sets but does not scale well for large data sets
  - Iterative computational complexity = $O(k(n-k)^2)$
  - Sampling method = CLARA

17

CLARA – sampling method

- Clustering LARge Applications
- Draw multiple samples (small portion of dataset D) of the data set
- Apply Partitioning Around Medoids on each sample to give the best clustering
- Perform better than PAM in larger data sets
- If selected in a random manner, it should closely represent the original dataset
- Efficiency depends on the sample size
  - A good clustering on samples may not be a good clustering of the whole data set

18

CLARANS

- Clustering Large Applications based upon RANdomized Search
- Since PAM uses entire dataset and CLARA can only select the medoids from the *chosen samples*
- CLARA cannot find the best clustering if any of the best sampled medoids is NOT among the best k-medoids chosen
- Draws a sample with some randomness in each step of the search

19

CLARANS

- Conceptually, clustering process can be seen as a search through a graph with each node as a potential solution
- 2 nodes are neighbors (connected by an arc) if their sets differ by only 1 object
- Each node is can be assigned a cost that is defined by the total dissimilarity of each object and the medoid of its cluster
- Partitioning Around Medoids (PAM) examines all of the current node's neighbors for a minimal cost solution

20

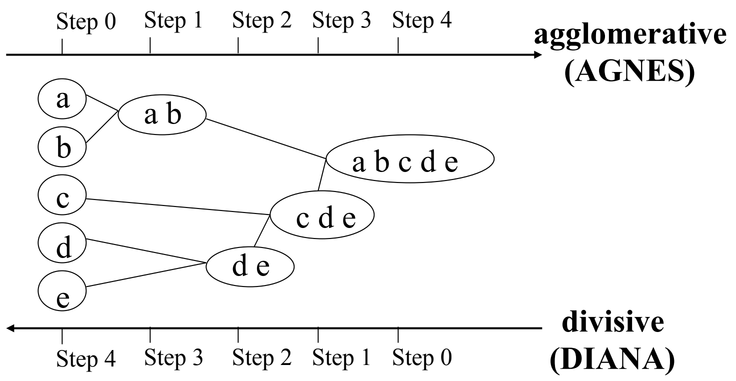
CLARANS (continued)

- CLARA works on a sample – has fewer neighbors to check
- CLARANS *dynamically* draws a random neighbor sample at each step of the process by not confining its search to *local* neighbors
- If a better neighbor, with a lower error is found, the process begins again
- Once the user-specified number of local minima has been found, the solution is output!

21

7.5 Hierarchical Clustering

- Group data objects into a tree of clusters



22

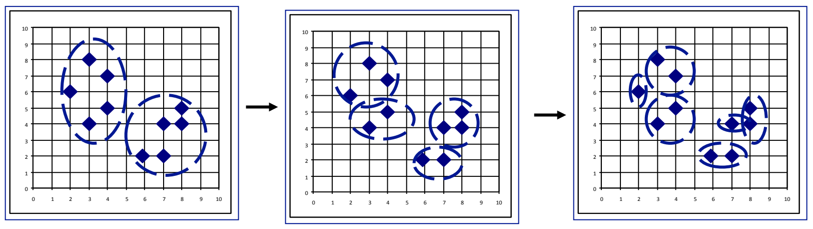
AGNES – Agglomerative NESting

- Initially, each object is a cluster
- Step-by-step cluster merging, until all objects form a cluster
  - Single-link approach
  - Each cluster is represented by all of the objects in the cluster
  - The similarity between two clusters is measured by the similarity of the closest pair of data points belonging to different clusters

23

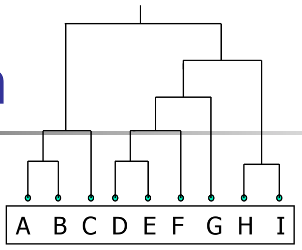
DIANA – Divisive ANAlysis

- Initially, all objects are in one cluster
- Step-by-step splitting clusters until each cluster contains only one object
  - Split according to some principle: the maximum Euclidean distance between the closest neighboring objects in the cluster.



24

Dendogram



- Dendogram →Tree structure to represent the process of hierarchical clustering
- Shows how to merge clusters hierarchically
- Decompose data objects into a multi-level nested partitioning (a tree of clusters)
- A clustering of the data objects: cutting the dendrogram at the desired level
  - Each connected component forms a cluster
  - {{{A,B},C},{{{{D,E},F},G},{H,I}}}

25

Distance Measures in Dendogram

- Single link: Minimum distance – nearest-neighbor clustering
- Complete link: Maximum distance – farthest-neighbor clustering
- Centroid: Mean distance
- Average linK: Average distance

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p,q)$$

$$d_{\max}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p,q)$$

$$d_{mean}(C_i, C_j) = d(m_i, m_j)$$

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{q \in C_j} d(p,q)$$

→m: mkan for a cluster
→C: a cluster
→n: the number of objects in a cluster

26

Hierarchical Clustering Challenges

- Hard to choose merge/split points
  - Cannot undo merging/splitting
  - Merging/splitting decisions are critical
- Computational complexity is high: $O(n^2)$
- Integrating hierarchical clustering with other techniques
  - BIRCH, ROCK, CHAMELEON

27

BIRCH - for numeric attributes

- **B**alanced **I**terative **R**educing and **C**lustering using **H**ierarchies
- Overcomes two difficulties of agglomerative clustering:
  - scalability
  - The inability to undo what was done in the previous step
- Introduces *clustering feature* and *clustering feature tree*
- CF (Clustering Feature) tree: a 3-D hierarchical data structure summarizing object information
  - Clustering objects → clustering leaf nodes of the CF tree
  - Summary of statistics for a given cluster

28

Clustering Feature Vector

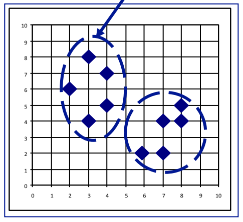
Clustering Feature: ***CF = (N, LS, SS)***

(data points) N: **Number of data points**

(Linear Sum) LS: $\sum^{N}_{i=1}=X_i$

(Square Sum) SS: $\sum^{N}_{i=1}=X_i^2$

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



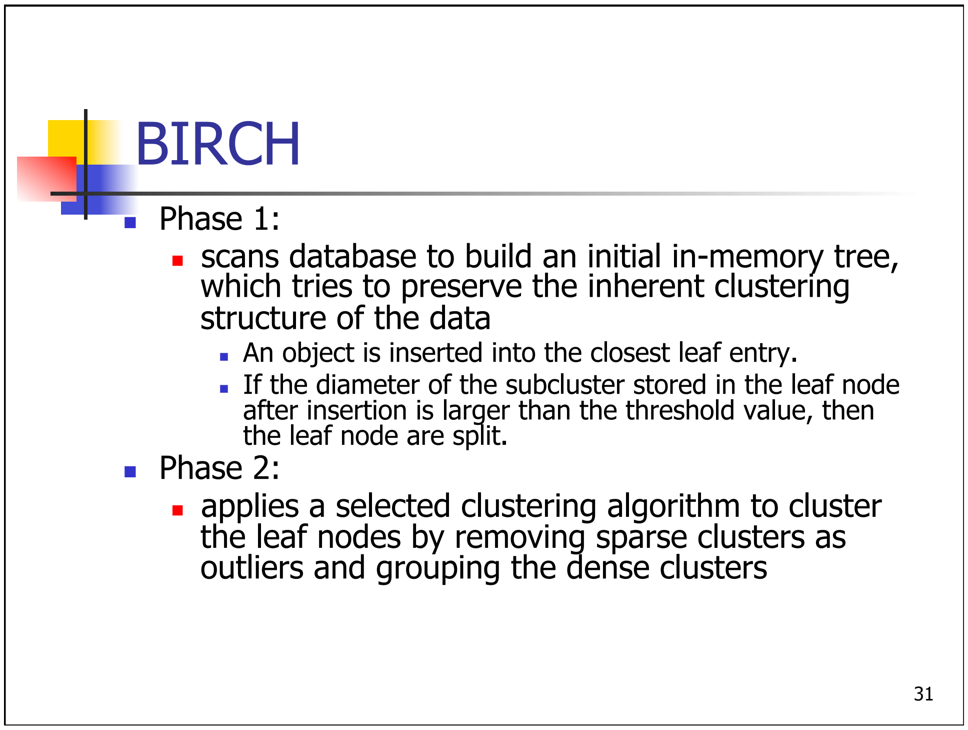
(3, 4)
(2, 6)
(4, 5)
(4, 7)
(3, 8)

29

Clustering Feature Vector

- These features are *additive*
- If CF1 has points (2,5), (3,2), (4,3)
- CF1 = {3, (2+3+4, 5+2+3), (2^2 + 3^2 + 4^2 , 5^2 + 2^2 + 3^2)} = {3,(9,10),(29,38)}
- Let CF2 = {3,(35,36),(417,440)}
- We add these to get →CF3 = {6,(44,46),(446,478)}

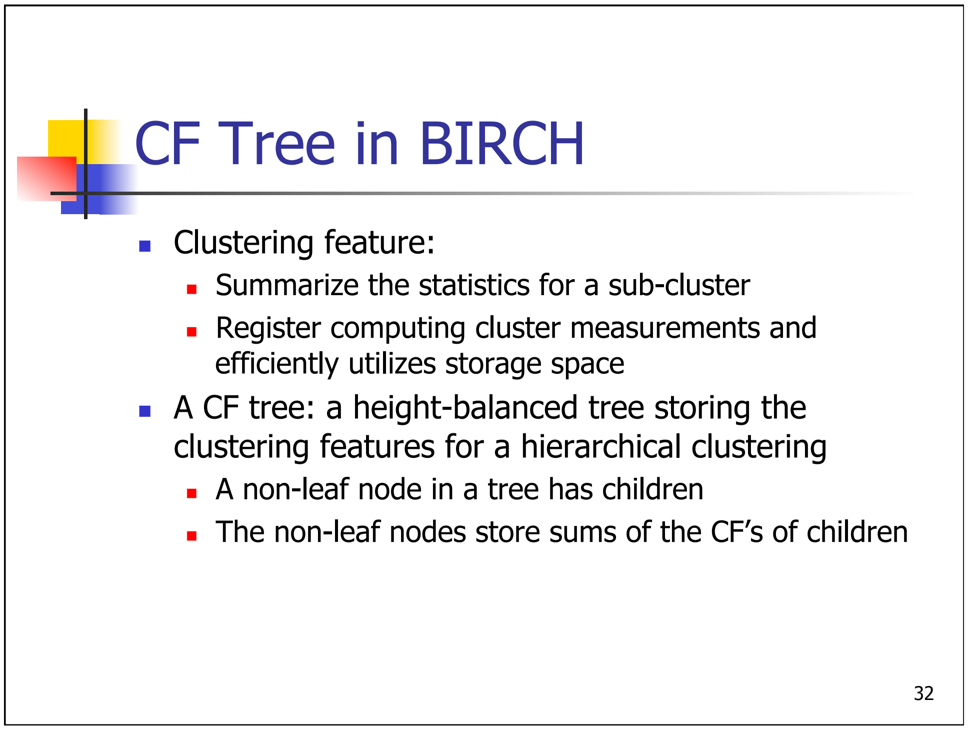
30



BIRCH

- Phase 1:
  - scans database to build an initial in-memory tree, which tries to preserve the inherent clustering structure of the data
    - An object is inserted into the closest leaf entry.
    - If the diameter of the subcluster stored in the leaf node after insertion is larger than the threshold value, then the leaf node are split.
- Phase 2:
  - applies a selected clustering algorithm to cluster the leaf nodes by removing sparse clusters as outliers and grouping the dense clusters

31

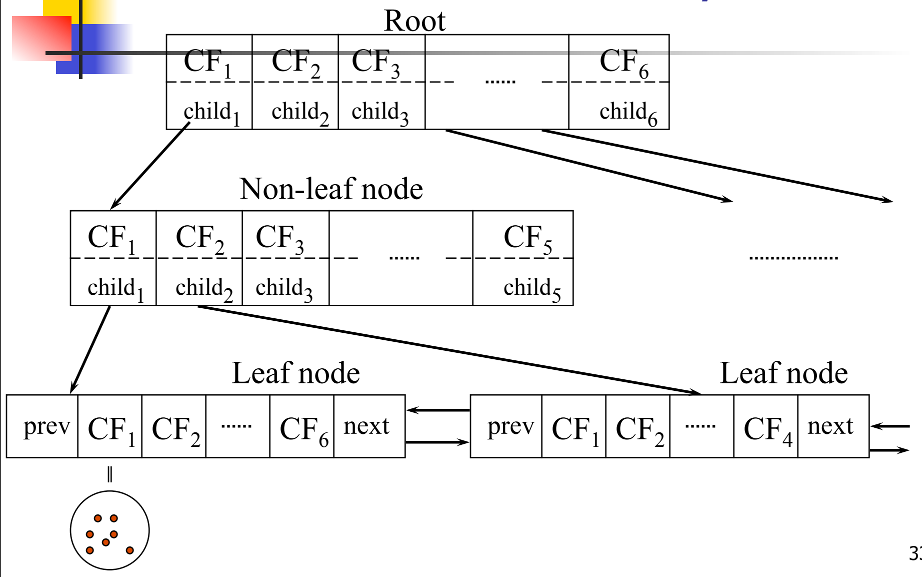


CF Tree in BIRCH

- Clustering feature:
  - Summarize the statistics for a sub-cluster
  - Register computing cluster measurements and efficiently utilizes storage space
- A CF tree: a height-balanced tree storing the clustering features for a hierarchical clustering
  - A non-leaf node in a tree has children
  - The non-leaf nodes store sums of the CF’s of children

32

CF Tree for cluster analysis



Root
CF_1 CF_2 CF_3 ...... CF_6
$child_1$ $child_2$ $child_3$ $child_6$
Non-leaf node
CF_1 CF_2 CF_3 ...... CF_5
$child_1$ $child_2$ $child_3$ $child_5$
....
Leaf node
prev CF_1 CF_2 ...... CF_6 next
Leaf node
prev CF_1 CF_2 ...... CF_4 next

33

CF Tree info

- Built dynamically as new data are inserted
- Used to guide a new insertion into the correct cluster
- A height-balanced tree
- 2 parameters:
  - branching factor *B* = max num of children for non-leaf node and
  - threshold *T* for = max diameter (or radius) of sub-clusters at the leaf nodes
- Each non-leaf node contains at most *B* entries
- Each leaf node has “prev” and “next” pointers to chain all leaf nodes together
- Diameter (or radius) of all entries in a leaf node must be less than *T*

34

Pros and Cons of BIRCH

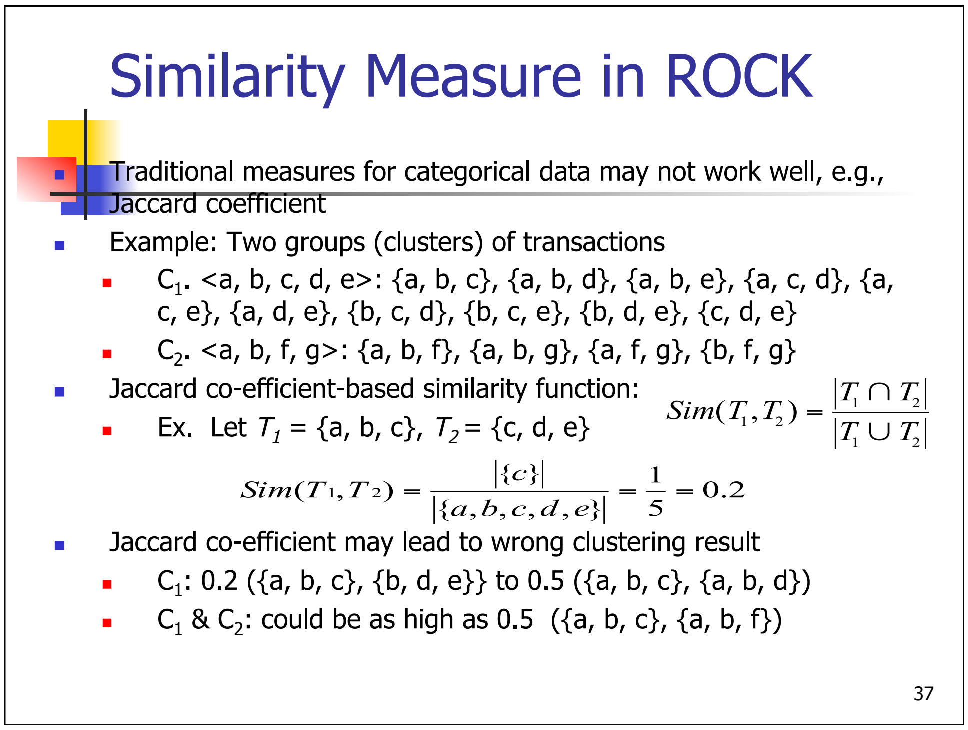
- Linear scalability
  - Good clustering with a single scan
- Can handle only numeric data
- Sensitive to the order of the data records
- Computational complexity of *O* (n)

35

Clustering Categorical Data: The ROCK Algorithm

- ROCK: **RO**bust **C**lustering using lin**K**s
  - S. Guha, R. Rastogi & K. Shim, ICDE' 99
- Basic Ideas:
  - If two data objects have similar neighborhoods, then the two data objects belong to the same cluster, and thus can be merged.
  - Use links (common neighbors between two objects) to measure similarity/proximity
  - Not distance-based

36



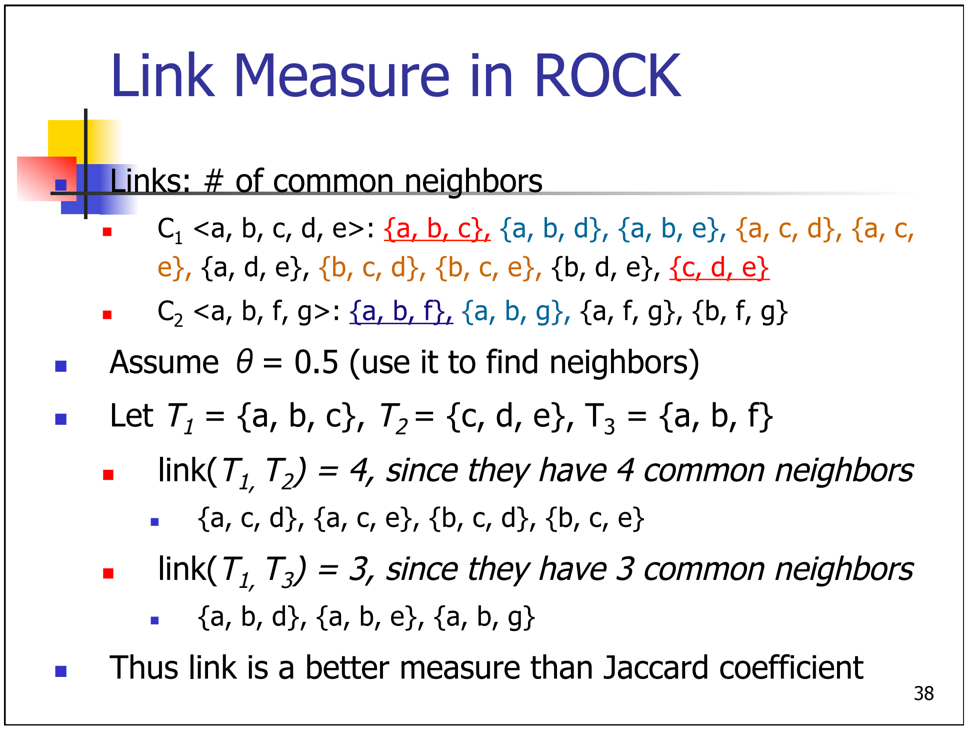
Similarity Measure in ROCK

- Traditional measures for categorical data may not work well, e.g., Jaccard coefficient
- Example: Two groups (clusters) of transactions
  - C_1. <a, b, c, d, e>: {a, b, c}, {a, b, d}, {a, b, e}, {a, c, d}, {a, c, e}, {a, d, e}, {b, c, d}, {b, c, e}, {b, d, e}, {c, d, e}
  - C_2. <a, b, f, g>: {a, b, f}, {a, b, g}, {a, f, g}, {b, f, g}
- Jaccard co-efficient-based similarity function: $Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$
  - Ex. Let T_1 = {a, b, c}, T_2 = {c, d, e}

$$Sim(T_1, T_2) = \frac{|\{c\}|}{|\{a, b, c, d, e\}|} = \frac{1}{5} = 0.2$$

- Jaccard co-efficient may lead to wrong clustering result
  - C_1: 0.2 ({a, b, c}, {b, d, e}} to 0.5 ({a, b, c}, {a, b, d})
  - C_1 & C_2: could be as high as 0.5 ({a, b, c}, {a, b, f})

37



Link Measure in ROCK

- Links: # of common neighbors
  - C_1 <a, b, c, d, e>: {a, b, c}, {a, b, d}, {a, b, e}, {a, c, d}, {a, c, e}, {a, d, e}, {b, c, d}, {b, c, e}, {b, d, e}, {c, d, e}
  - C_2 <a, b, f, g>: {a, b, f}, {a, b, g}, {a, f, g}, {b, f, g}
- Assume θ = 0.5 (use it to find neighbors)
- Let T_1 = {a, b, c}, T_2 = {c, d, e}, T_3 = {a, b, f}
  - link(T_1, T_2) = 4, *since they have 4 common neighbors*
    - {a, c, d}, {a, c, e}, {b, c, d}, {b, c, e}
  - link(T_1, T_3) = 3, *since they have 3 common neighbors*
    - {a, b, d}, {a, b, e}, {a, b, g}
- Thus link is a better measure than Jaccard coefficient

38

7.6 Density-Based Clustering Methods

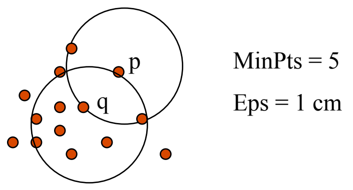
- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition

39

Density-Based Clustering: Basic Concepts

- Two parameters*:*
  - *Eps*: Maximum radius of the neighbourhood
  - *MinPts*: Minimum number of points in an Eps-neighbourhood of that point
- $N_{Eps}(p)$: $\{q \text{ belongs to } D \mid dist(p,q) \le Eps\}$
- Directly density-reachable: A point *p* is directly density-reachable from a point *q* w.r.t. *Eps, MinPts* if
  - *p* belongs to $N_{Eps}(q)$
  - core point condition:

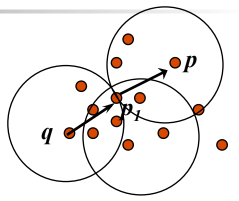
    $|N_{Eps}(q)| \ge MinPts$



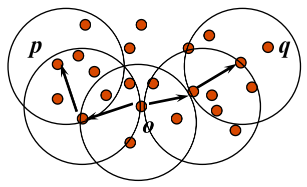
40

Density-Reachable and Density-Connected

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



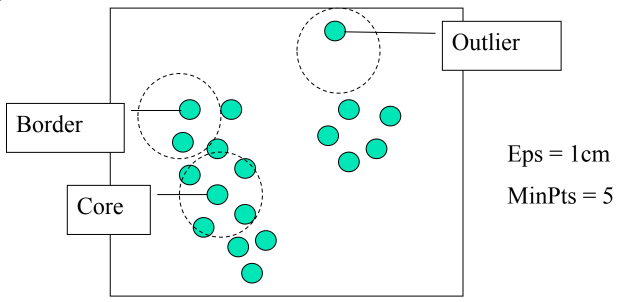
- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



41

DBSCAN: **D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



42

DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. *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.

43