# Clustering

## Learning Outcomes

- Understand Clustering Analysis
- Types of Data in Clustering Analysis
- Major Clustering Methods
- Partitioning Methods
- Hierarchical Methods

## Cluster Analysis

- Cluster
  - Collection of data objects similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Clustering
  - Grouping a set of data objects into clusters
  - is unsupervised classification
  - no predefined classes
- Typical applications
  - Stand-alone tool to get insight into data distribution
  - Pre-processing step for other algorithms

## Clustering Applications

- Pattern Recognition
- Spatial Data Analysis
  - create thematic maps in GIS by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- Image Processing
- Economic Science (especially market research)
- WWW
  - Document classification
  - Cluster Weblog data to discover groups of similar access patterns

## Examples of Clustering Applications

- Marketing
  - Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Insurance
  - Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning
  - Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies
  - Observed earth quake epicentres should be clustered along continent faults

## Good Clustering

- A good clustering method will produce high quality clusters with
  - high intra-cluster similarity
  - low inter-cluster similarity
- Clustering quality
  - depends on both the similarity measure used by the method and its implementation
  - is also measured by its ability to discover some or all of the hidden patterns

## Clustering Requirements

- Scalability
- Ability to deal with
  - different types of attributes
  - Noise and outliers
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Insensitive to order of input records
- Incorporation of user-specified constraints
- Interpretability and usability

## Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, which is typically metric: $d(i, j)$
- The definitions of distance functions are usually very different for interval-scaled, Boolean, categorical, ordinal and ratio variables
- Weights should be associated with different variables based on applications and data semantics
- It is hard to define “similar enough” or “good enough”
  - the answer is typically highly subjective

## Types of data in clustering analysis

- Interval-scaled variables
- Binary variables
- Nominal variables
- Ordinal variables
- Ratio-scaled variables
- Variables of mixed types

## Interval-valued variables

- Standardise dataType equation here.
  - Calculate the mean absolute deviation:

$$S_a = \frac{1}{n}(|x_1 - m_a| + |x_2 - m_a| + \cdots + |x_n - m_a|)$$

$$m_a = \frac{1}{n}(x_1 + x_2 + \ ... + x_n)$$

where

  - Calculate the standardised measurement (*z-score*)

$$z_{ia} = \frac{x_i - m_a}{s_a}$$

- Mean absolute deviation is more robust than using standard deviation

## Similarity & Dissimilarity Between Objects

- Distance
  - is normally used to measure the similarity or dissimilarity between two data objects
- Some popular ones include
  - *Minkowski distance*:

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{in} - x_{jn}|^q)}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ are two $n$-dimensional data objects, and $q$ is a positive integer

- Manhattan distance ( $q = 1$)

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}|$$

## Similarity & Dissimilarity Between Objects

- Euclidean distance ($q = 2$)

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{in} - x_{jn}|^2)}$$

  - Properties
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$
- Also one can use weighted distance, parametric Pearson product moment correlation, or other dissimilarity measures

# Binary Variables

- A contingency table for binary data

| Object *i* \ Object *j* | 1 | 0 | *sum* |
|---|---|---|---|
| 1 | $a$ | $b$ | $a+b$ |
| 0 | $c$ | $d$ | $c+d$ |
| *sum* | $a+c$ | $b+d$ | $p$ |

- Simple matching coefficient (invariant, if the binary variable is *symmetric*): $d(i,j) = \frac{b+c}{a+b+c+d}$
- Jaccard coefficient (non invariant if the binary variable is *asymmetric*): $d(i,j) = \frac{b+c}{a+b+c}$

# Dissimilarity between Binary Variables

- Example

| Name | Gender | Fever | Cough | Test-1 | Test-2 | Test-3 | Test-4 |
|---|---|---|---|---|---|---|---|
| Jack | M | Y | N | P | N | N | N |
| Mary | F | Y | N | P | N | P | N |
| Jim | M | Y | P | N | N | N | N |

- gender is a symmetric attribute
- the remaining attributes are asymmetric binary
- let the values Y and P be set to 1, and the value N be set to 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$
$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$
$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

# Nominal Variables

- Categorical variables
  - Can take more than 2 states, e.g., red, yellow, blue, green
- Method 1
  - Simple matching
  - *M*: number of matches,
  - P: total number of variables

$$d(i,j) = \frac{P - M}{P}$$

- Method 2
  - use a large number of binary variables
  - creating a new binary variable for each of the *M* nominal states

# Ordinal Variables

- An ordinal variable can be discrete or continuous
- The order is important, e.g., rank
- Can be treated like interval-scaled
  - replacing $x_{ij}$ by their rank $r_{ij} \in \{1, \dots, M_j\}$
  - map the range of each variable onto [0, 1] by replacing $i^{th}$ object in the $j^{th}$ variable by

$$z_{ij} = \frac{r_{ij} - 1}{M_j - 1}$$

  - Compute the dissimilarity using methods for interval-scaled variables

# Ratio-Scaled Variables

- Ratio-scaled variable
  - a positive measurement on a nonlinear scale, approximately at exponential scale
  - E.g. $Ae^{Bt}$ or $Ae^{-Bt}$
- Methods
  - treat them like interval-scaled variables — *not a good choice!*
  - apply logarithmic transformation

$$y_i = \log(x_i)$$

  - treat them as continuous ordinal data and treat their rank as interval-scaled

# Variables of Mixed Types

- A database may contain all the six types of variables
  - symmetric binary, asymmetric binary, nominal, ordinal, interval and ratio
- One may use a weighted formula to combine their effects

$$d(i,j) = \frac{\sum_{k=1}^{P} \delta_{ij}^{k} d_{ij}^{k}}{\sum_{k=1}^{P} \delta_{ij}^{k}}$$

  - $f$ is binary or nominal:

    $\delta_{ij}^{k} = 0$ if $x_{ik} = x_{jk}$, or $\delta_{ij}^{k} = 1$ othewise
  - $f$ is interval-based: use the normalised distance
  - $f$ is ordinal or ratio-scaled
    - compute ranks $r_{ij}$ and
    - and treat $z_{ij}$ as interval-scaled

$$z_{ij} = \frac{r_{ij} - 1}{M_j - 1}$$

## Major Clustering Techniques

- Partitioning algorithms
  - Construct various partitions and then evaluate them by some criterion
- Hierarchy algorithms
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
- Density-based
  - based on connectivity and density functions
- Grid-based
  - based on a multiple-level granularity structure
- Model-based
  - A model is hypothesised for each of the clusters and the idea is to find the best fit of that model to each other

## Partitioning Algorithms: Basic Concept

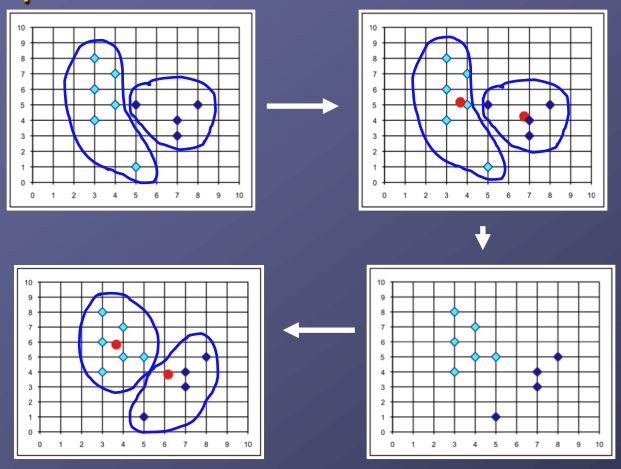
- Partitioning method
  - Partition a database ***D*** of ***n*** objects into a set of ***k*** clusters
- Problem
  - Given ***k***, find a partition of ***k*** *clusters* that optimises the chosen partitioning criterion
- Remarks
  - Global optimal: exhaustively enumerate all partitions (!!!)
  - Heuristic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

# The *K-Means* Clustering Method

- *k-means* algorithm is implemented in 4 steps
- Step 1
  - Given k, partition objects into *k* nonempty subsets
- Step 2
  - Compute seed points as the centroids of the clusters of the current partition
  - The centroid is the centre (mean point) of the cluster
- Step 3
  - Assign each object to the cluster with the nearest seed point
- Step 4
  - Go back to Step 2 until no more new assignment

# The *K-Means* Clustering Method

- Example



## *K-Means* Method

- Strength
  - *Relatively efficient*: $O(tkn)$, where $n, k,$ and $t$ are the number of objects, number of clusters, and number iterations respectively. Normally, $k, t \ll n$.
  - Often terminates at a *local optimum*
  - The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

## Variations of the *K-Means* Method

- A few variants of the *k-means* differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

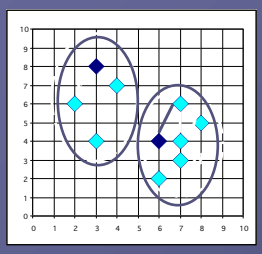
## The *K-Medoids* Clustering Method

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids)
  - Start from an initial set of medoids and iteratively replace one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- Focusing + spatial data structure

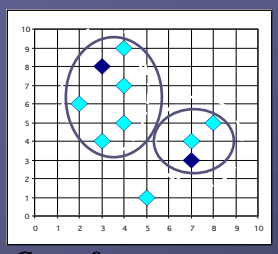
## PAM (Partitioning Around Medoids)

- Use real object to represent the cluster
  - Select ***k*** representative objects arbitrarily
  - For each pair of non-selected object ***h*** and selected object ***i***, calculate the total swapping cost $TC_{ih}$
  - For each pair of ***i*** and ***h***,
    - If $TC_{ih} < 0$, ***i*** is replaced by ***h***
    - Then assign each non-selected object to the most similar representative object
  - Repeat steps 2-3 until there is no change

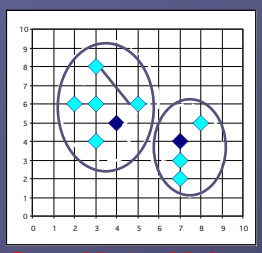
## PAM Clustering: Total swapping cost $TC_{ih}=\sum_j C_{jih}$



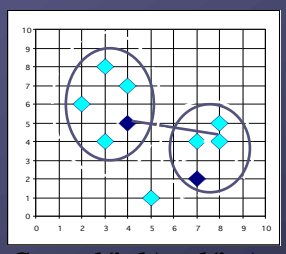
$C_{jih} = d(j, h) - d(j, i)$



$C_{jih} = 0$



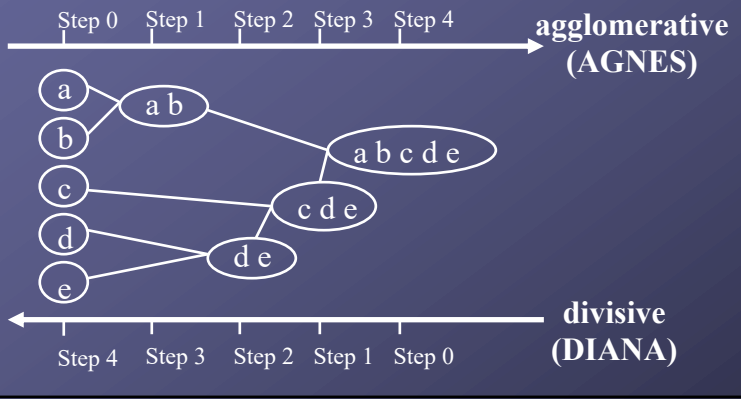
$C_{jih} = d(j, t) - d(j, i)$



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

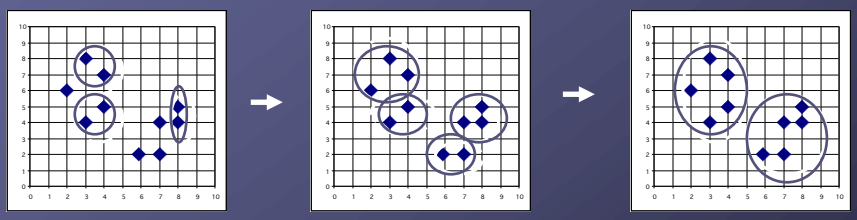
## Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



# AGNES (Agglomerative Nesting)

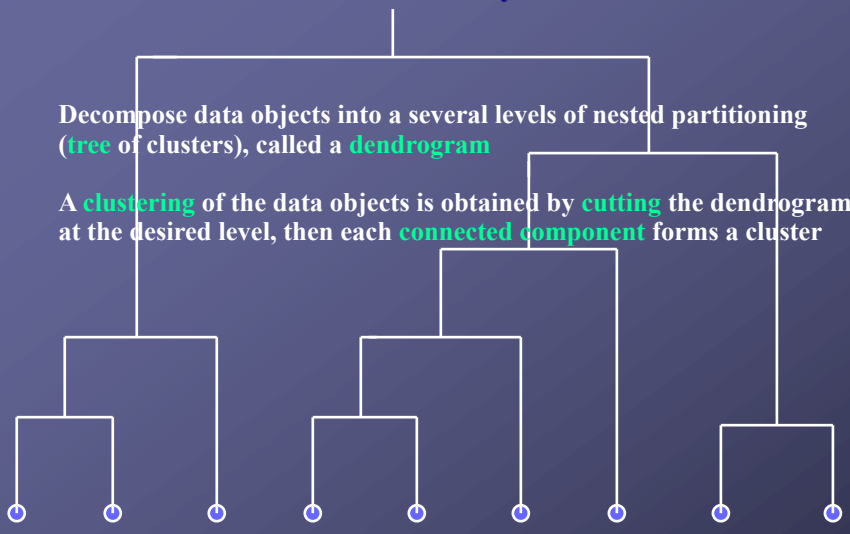
- Use the Single-Link method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Continue in a non-descending fashion
- Eventually all nodes belong to the same cluster



## A *Dendrogram* Shows How the Clusters are Merged Hierarchically

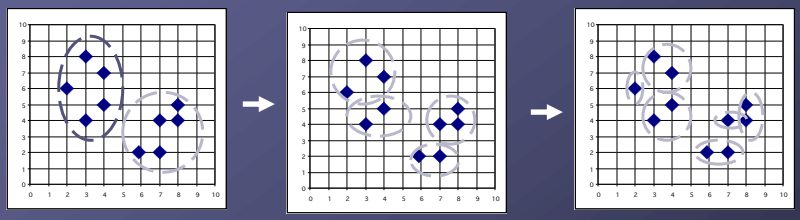
Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster



## DIANA (Divisive Analysis)

- Inverse order of AGNES
- Eventually each node forms a cluster on its own



## Hierarchical Clustering Methods

- Major weakness of agglomerative clustering methods
  - do not scale well: time complexity of at least $O(n^2)$, where $n$ is the total number of objects
  - can never undo what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH: uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CURE: selects well-scattered points from the cluster and then shrinks them towards the centre of the cluster by a specified fraction

# Clustering Feature Vector

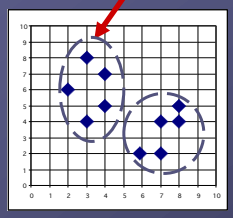
**Clustering Feature:** $CF = (N, \vec{LS}, SS)$

$N$: **Number of data points**

$LS$: $\sum_{i=1}^{N} \vec{X_i}$ : sum of the points in the cluster

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

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

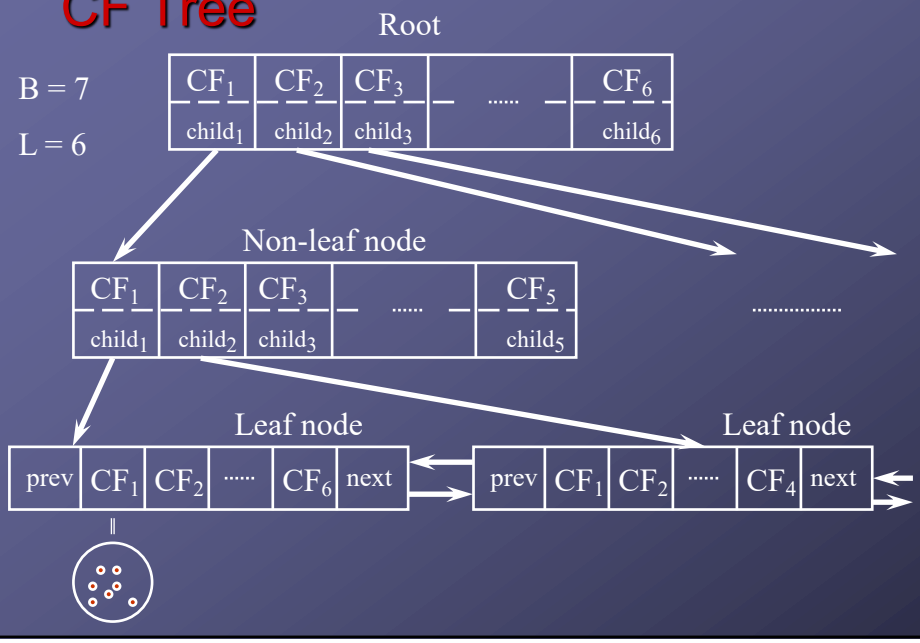


(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

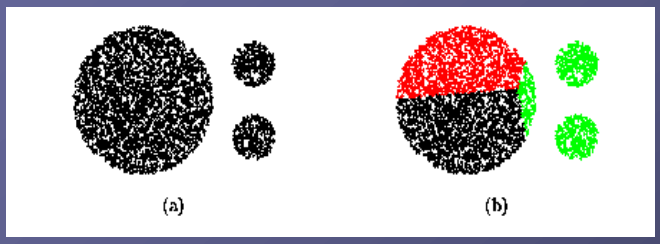
# CF Tree

B = 7

L = 6

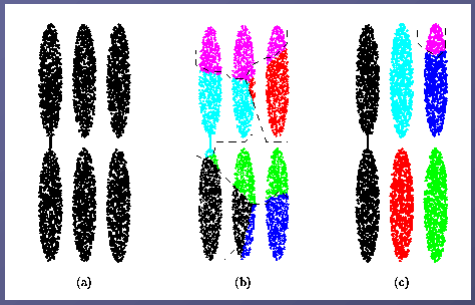


# CURE (Clustering Using Representatives)




- The main idea of CURE
  - Stops the creation of a cluster hierarchy if a level consists of *k* clusters
  - Uses multiple representative points to evaluate the distance between clusters, adjusts well to arbitrary shaped clusters and avoids single-link effect

# Drawbacks of Distance-Based Method



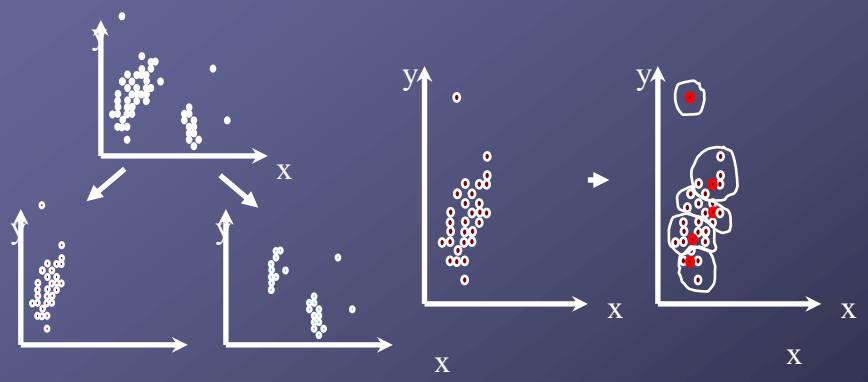

- Drawbacks of square-error based clustering method
  - Consider only one point as representative of a cluster
  - Good only for convex shaped, similar size and density, and if *k* can be reasonably estimated

# Cure: The Algorithm

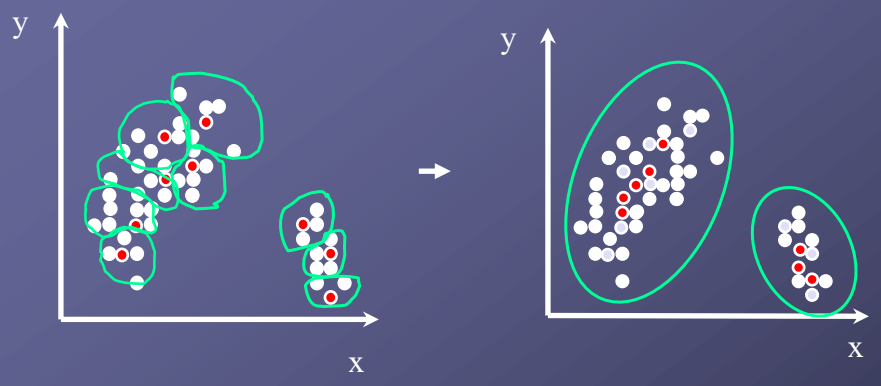
- Draw random sample *s*
- Partition sample to *p* partitions with size *s/p*
- Partially cluster partitions into *s/pq* clusters
- Eliminate outliers
  - By random sampling
  - If a cluster grows too slow, eliminate it
- Cluster partial clusters
- Label data in disk

# Data Partitioning and Clustering

- s = 50
- p = 2
- s/p = 25
- s/pq = 5



## Cure: Shrinking Representative Points




- Shrink the multiple representative points towards the gravity centre by a fraction of $\alpha$.
- Multiple representatives capture the shape of the cluster

## Clustering Categorical Data: ROCK

- ROCK: Robust Clustering using linKs
  - Use links to measure similarity/proximity
  - Not distance based
  - Computational complexity:

$$O(n^2 + nm_m m_a + n^2 \log n)$$

- Basic ideas
  - Similarity function and neighbours:

Let $T_1$ = {1,2,3}, $T_2$={3,4,5}

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

$$Sim(T1, T2) = \frac{|\{3\}|}{|\{1,2,3,4,5\}|} = \frac{1}{5} = 0.2$$

## Rock Algorithm

- Links: The number of common neighbours for the two points

{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}
{1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}

{1,2,3} ⟷ (3) {1,2,4}

- Algorithm
  - Draw random sample (get a random sample of the data)
  - Cluster the data using the link agglomerative approach
  - Label data in disk

## Example

- Consider a system where documents may contain keywords {book, water, sun, sand, swim, read}. Suppose there are 4 documents, where
  - 1st contains the word {book},
  - 2nd contains {water, sun, sand, swim},
  - 3rd contains {water, sun, swim, read}, and
  - 4th contains {read, sand}.
- Use the ROCK clustering algorithm to cluster these documents.

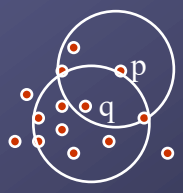
## Density-Based Clustering Methods

- 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

## Density-Based Clustering: Background

- Two parameters:
  - ***Eps***: Maximum radius of the neighbourhood
  - ***MinPts***: Minimum number of points in an Eps-neighbourhood of that point
- $N_{Eps}(p)$: *{q belongs to D | dist(p,q) <= Eps}*
- Directly density-reachable: A point ***p*** is directly density-reachable from a point ***q*** with regard to ***Eps***, ***MinPts*** if
  - 1) ***p*** belongs to $N_{Eps}(q)$
  - 2) core point condition:

    $|N_{Eps}(q)| >= MinPts$



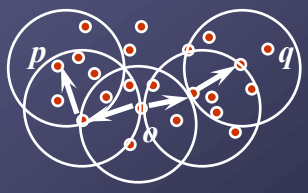

MinPts = 5

Eps = 1 cm

## Density-Based Clustering: Background (II)

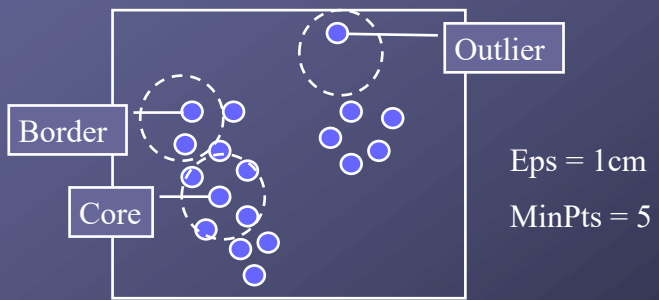
- Density-reachable:
  - A point *p* is density-reachable from a point *q* with regard to *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that $p_{i+1}$ is directly density-reachable from $p_i$


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




## DBSCAN: Density Based Spatial Clustering of Applications with Noise

- 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



# DBSCAN: The Algorithm

- Arbitrary select a point ***p***
- Retrieve all points density-reachable from ***p*** with regard to ***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.