clustering

What is Cluster Analysis?

- **Cluster: a collection of data objects**
  - **Similar to one another within the same cluster**
  - **Dissimilar to the objects in other clusters**
- **Cluster analysis**
  - **Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters**
- **Unsupervised learning: no predefined classes**
- **Typical applications**
  - **As a stand-alone tool to get insight into data distribution**
  - **As a preprocessing step for other algorithms**

Clustering: Rich Applications

- **Pattern Recognition**
- **Image Processing**
- **Economic Science (especially market research)**
- **Business: customer group based on purchasing pattern**
- **Used for outlier detection**
- **WWW**
  - **Document classification**
  - **Cluster Weblog data to discover groups of similar access patterns**

Requirements of Clustering in Data Mining

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Incremental clustering and Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints (e.g., choose location for new ATM of a bank)
- Interpretability and usability

Data Structures

- Data matrix (object by variable structure)
  - (two modes) The rows and columns of the data matrix represent different entities

attributes/dimensions

tuples/objects

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- Dissimilarity matrix (object by object structure)
  - (one mode) dissimilarity matrix represent the same entity

objects

objects

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Interval-scaled variables

- Interval-scaled variables are continuous measurements of a roughly linear scale. E.g. - weight and height, latitude and longitude coordinates (e.g., when clustering houses), and weather temperature.
- measurement unit - affect the clustering analysis
- Standardize data - To avoid dependence on the choice of measurement units
- some applications - users may intentionally want to give more weight to a certain set of variables than to others.
- E.g. clustering basketball player, more weight to the variable height
- *How can the data for a variable be standardized?"*
  - to convert the original measurements to unitless variables

Interval-scaled variables

1. Calculate the mean absolute deviation for a variable *f*:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$$

where

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf})$$

- The mean absolute deviation, s_f, is more robust to outliers than the standard deviation, *s* (σ)

2. Calculate the standardized measurement (*z-score*)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- Standardization may or may not be useful in a particular application.
- Choice of whether and how to perform standardization should be left to the user
- Distance measures: Euclidean, Manhattan, Minkowski

Given the following measurements for the variable *age*:18, 22, 25, 42, 28, 43, 33, 35, 56, 28

standardize the variable by the following:

(a) Compute the mean absolute deviation of *age*.

(b) Compute the z-score for the first four measurements.

- m_f=(18+22+25+42+28+43+33+35+56+28)/10
- =330/10=33
- s_f=(|18-33|+|22-33|+|25-33|+|42-33|+|28-33|+|43-33|+|33-33|+ 35-33|+56-33|+|28-33|)/10
- =(15+11+8+9+5+10+0+2+23+5)/10
- =88/10=8.8

$$z_{if} = \frac{x_{if} - m_{if}}{s_f}$$

$$z_{1f} = \frac{18 - 33}{8.8} = -1.70$$
$$z_{2f} = \frac{22 - 33}{8.8} = -1.25$$
$$z_{3f} = \frac{25 - 33}{8.8} = -0.91$$
$$z_{4f} = \frac{42 - 33}{8.8} = 1.02$$

Distance Measures

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

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

- **Manhattan (city block) distance** $d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}|$.

1. $d(i, j) \geq 0$: Distance is a nonnegative number.
2. $d(i, i) = 0$: The distance of an object to itself is 0.
3. $d(i, j) = d(j, i)$: Distance is a symmetric function.
4. $d(i, j) \leq d(i, h) + d(h, j)$: Going directly from object i to object j in space is no more than making a detour over any other object h (*triangular inequality*).

Distance Measures

- **Minkowski distance** $$d(i, j) = (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \cdots + |x_{in} - x_{jn}|^p)^{1/p}$$

where p is a positive integer

Generalization of both Euclidean and Manhattan distance

also called *Lp* norm, in some literature.It represents the Manhattan distance when $p = 1$ (i.e., L_1 norm) and Euclidean distance when $p = 2$ (i.e., L_2 norm).

If each variable is assigned a weight according to its perceived importance, the weighted Euclidean distance can be computed as

$$d(i, j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \cdots + w_m|x_{in} - x_{jn}|^2}.$$

Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8):

(a) Compute the *Euclidean distance* between the two objects.

(b) Compute the *Manhattan distance* between the two objects.

(c) Compute the *Minkowski distance* between the two objects, using $p = 3$.

(a) Compute the *Euclidean distance* between the two objects.

$$\begin{aligned} d(i,j) &= \sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+\cdots+(x_{in}-x_{jn})^2} \\ &= \sqrt{|22-20|^2+|1-0|^2+|42-36|^2+|10-8|^2} = 6.71 \end{aligned}$$

(b) Compute the *Manhattan distance* between the two objects.

$$\begin{aligned} d(i,j) &= |x_{i1}-x_{j1}|+|x_{i2}-x_{j2}|+\cdots+|x_{in}-x_{jn}| \\ &= |22-20|+|1-0|+|42-36|+|10-8| = 11 \end{aligned}$$

(c) Compute the *Minkowski distance* between the two objects, using $p = 3$.

$$\begin{aligned} d(i,j) &= (|x_{i1}-x_{j1}|^p+|x_{i2}-x_{j2}|^p+\cdots+|x_{in}-x_{jn}|^p)^{1/p} \\ &= (|22-20|^3+|1-0|^3+|42-36|^3+|10-8|^3)^{1/3} = 6.15 \end{aligned}$$

Binary Variables

- A contingency table for binary data
- Distance measure for symmetric binary variables:
- A binary variable is symmetric if both of its states are equally valuable and carry the same weight
- E.g. gender (M/F)

		Object j		
		1	0	*sum*
Object i	1	q	r	$q+r$
	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

Binary Variables

- Distance measure for asymmetric binary variables:
- A binary variable is asymmetric if the outcomes of the states are not equally important
- E.g. *positive* and *negative* outcomes of a disease *test*

$$d(i, j) = \frac{r + s}{q + r + s}$$

- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

$$sim_{Jaccard}(i, j) = \frac{q}{q + r + s} = 1 - d(i, j).$$

Dissimilarity between Binary Variables

- **Example**

Name	Fever	Cough	Test-1	Test-2	Test-3	Test-4
P1	Y	N	P	N	N	N
P2	Y	N	P	N	P	N
P3	Y	P	N	N	N	N

  - **the attributes are asymmetric binary**
  - **let the values Y and P be set to 1, and the value N be set to 0**

$$d(P1, P2) = \frac{0+1}{2+0+1} = 0.33$$

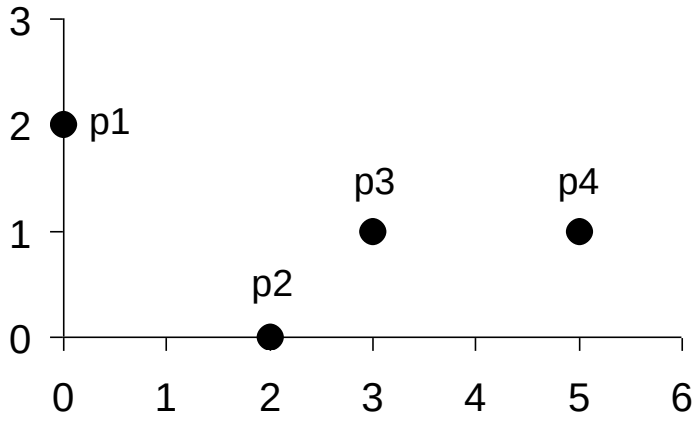
$$d(P1, P3) = \frac{1+1}{1+1+1} = 0.67$$

$$d(P3, P2) = \frac{1+2}{1+1+2} = 0.75$$

$$d(i, j) = \frac{r+s}{q+r+s}$$

q:11 r:10
s:01 t:00

Example: Data Matrix and Distance Matrix



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Data Matrix

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix (i.e., Dissimilarity Matrix) for Euclidean Distance

Categorical Variables

- A generalization of the binary variable in that it can take more than 2 states, e.g., map_color(red, yellow, blue, green)
- Method 1: Simple matching
  - *m*: # of matches, *p*: total # of variables

$$d(i, j) = \frac{p - m}{p}$$

object identifier	test-1 (categorical)
1	code-A
2	code-B
3	code-C
4	code-A

P=1

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix}$$

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank, designation, medals
- Can be treated like interval-scaled
  - replace x_{if} by their rank $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

  - compute the dissimilarity using methods for interval-scaled variables

object identifier	test-2 (ordinal)	
1	excellent	3
2	fair	1
3	good	2
4	excellent	3

$\{r_{if}=1,2,3\}$

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

fair=(1-1)/(3-1)=0
good=(2-1)/(3-1)=0.5
excellent=(3-1)/(3-1)=1

- ***M_f=3. fair, good, excellent***
- ***Normalize – 0.0 to 1.0***
- ***Euclidean distance***

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

Ratio-Scaled Variables

- **Ratio-scaled variable: a positive measurement on a nonlinear scale, approximately at exponential scale, such as Ae^{Bt} or Ae^{-Bt}**
- **3 Methods:**
- **treat them like interval-scaled variables—*not a good choice!* (why?—the scale can be distorted)**
- **apply logarithmic transformation (to a ratio-scaled variable *f* having value x_{if} for object *i* by using the formula)**

$$y_{if} = log(x_{if})$$

The y_{if} values can be treated as interval scaled variables

- **treat them as continuous ordinal data, treat their rank as interval-scaled**

object identifier	test-3 (ratio-scaled)
1	445
2	22
3	164
4	1,210

$$\begin{bmatrix} 0 & & & \\ 1.31 & 0 & & \\ 0.44 & 0.87 & 0 & \\ 0.43 & 1.74 & 0.87 & 0 \end{bmatrix}$$

- **Taking the *log* of *test-3* results in the following values: log(445)= 2.65, log(22)=1.34, log(164)=2.21, and log(1210)=3.08 for the objects 1 to 4, respectively.**
- **Use Euclidean distance**
- **$D(2,1)=\sqrt{(1.34-2.65)^2} = 1.31$**
- **$D(3,1)=\sqrt{2.21-2.65)^2} = 0.44$**

Vector Objects

- **Vector objects: keywords in documents, gene features in micro-arrays, etc.**
- **Broad applications: information retrieval, biologic taxonomy, etc.**
- **Cosine measure** $s(x, y) = \frac{x^t \cdot y}{||x|| ||y||}$

x^t is a transposition of vector x, $||x||$ is the Euclidean norm of vector x defined as $\sqrt{x_1^2 + x_2^2 + \ldots + x_p^2}$.

given two vectors, *x* = (1, 1, 0, 0) and *y* = (0, 1, 1, 0) find the similarity between x & y by using cosine distance

$$x^t = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad y = [\,0\ 1\ 1\ 0\,]$$

$$s(x, y) = \frac{(0+1+0+0)}{\sqrt{2}\sqrt{2}} = 0.5$$

Example:

d_1 = 3 2 0 5 0 0 0 2 0 0

d_2 = 1 0 0 0 0 0 0 1 0 2

$d_1 \bullet d_2 = 3*1+2*0+0*0+5*0+0*0+0*0+0*0+2*1+0*0+0*2 = 5$

$\|d_1\| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$

$\|d_2\| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5} = (6)^{0.5} = 2.245$

$\cos(d_1, d_2) = 5/(6.481*2.245) = 0.34$

Major Clustering Approaches (I)

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, ROCK, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue

Major Clustering Approaches (II)

- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE
- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: pCluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering

Partitioning Algorithms: Basic Concept

- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters (k <= n)
- Iterative optimization paradigm
- 2 main categories
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition Around Medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

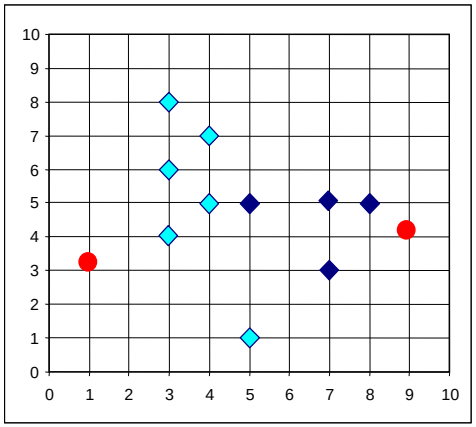
Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) repeat
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) until no change;

The *K-Means* Clustering Method

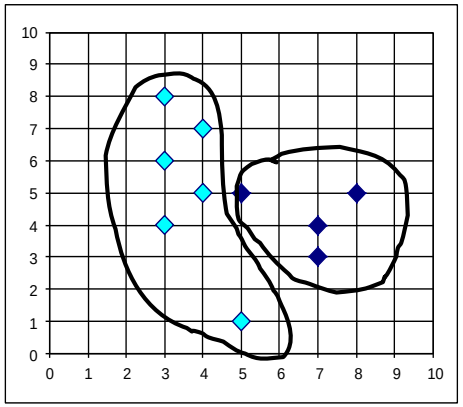
- Example



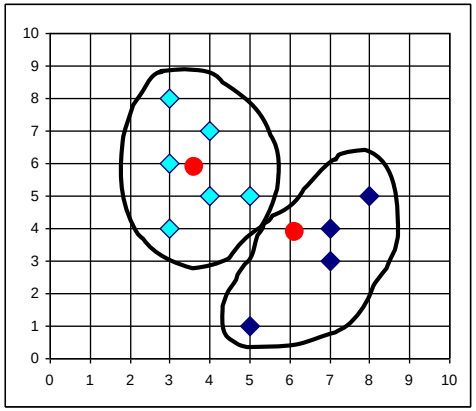
K=2

Arbitrarily choose K object as initial cluster center

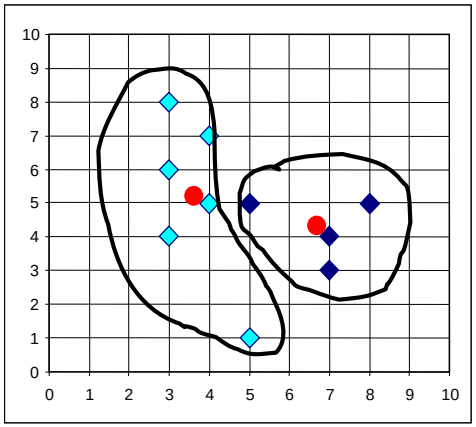
Assign each objects to most similar center



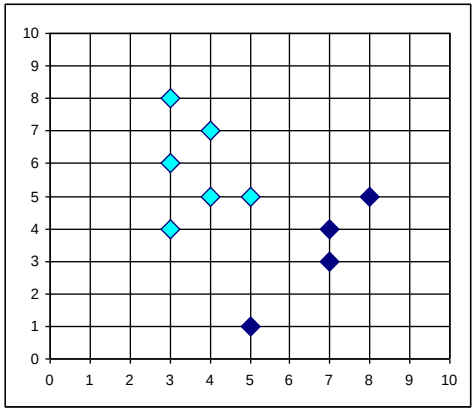
reassign



Update the cluster means



reassign



Update the cluster means

Distance measures

distance measures that are commonly used for computing the dissimilarity of objects:

- Euclidean distance
- Manhattan distance
- Minkowski distance

Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where m_i is the centroid or medoid of cluster C_i)

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - m_i)^2$$

K-Means Example

- Given: {2,4,10,12,3,20,30,11,25}, k=2
- Randomly assign means: $m_1=3, m_2=4$
- $K_1=\{2,3\}$, $K_2=\{4,10,12,20,30,11,25\}$, E=1523
- $m_1=(2+3)/2=2.5, m_2=112/7=16$
  $K_1=\{2,3,4\}, K_2=\{10,12,20,30,11,25\}$, E=372.75
  $m_1=3, m_2=18$
- $K_1=\{2,3,4,10\}, K_2=\{12,20,30,11,25\}$, E=284
  $m_1=4.75, m_2=19.6$
- $K_1=\{2,3,4,10,11,12\}, K_2=\{20,30,25\}$, E=267.85
  $m_1=7, m_2=25$
- Stop as the clusters with these means are the same.

cluster the following points into 3 clusters by k-means method *A*1(2, 10); *A*2(2, 5); *A*3(8, 4); *B*1(5,8); *B*2(7, 5); *B*3(6, 4); *C*1(1, 2); *C*2(4, 9).

NOTE:

- The distance function is Euclidean distance.
- initially assign *A*1, *B*1, and *C*1 as the center of each cluster, respectively.

		(2, 10)	(5,8)	(1,2)	Cluster
A1	(2, 10)	0	3.605551	8.062258	CLUST1
A2	(2, 5)	5	4.242641	3.162278	CLUST3
A3	(8, 4)	8.485281	5	7.28011	CLUST2
B1	(5, 8)	3.605551	0	7.211103	CLUST2
B2	(7, 5)	7.071068	3.605551	6.708204	CLUST2
B3	(6, 4)	7.211103	4.123106	5.385165	CLUST2
C1	(1, 2)	8.062258	7.211103	0	CLUST3
C2	(4, 9)	2.236068	1.414214	7.615773	CLUST2

CLUST1={A1}
CLUST2={A3,B1,B2,B3,C2}
CLUST3={A2,C1}

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

New cluster centers
CLUST1=(2,10)
CLUST2=((8+5+7+6+4)/5, (4+8+5+4+9)/5)=(6,6)
CLUST3=((2+1)/2, (5+2)/2) = (1.5,3.5)

Comments on the *K-Means* Method

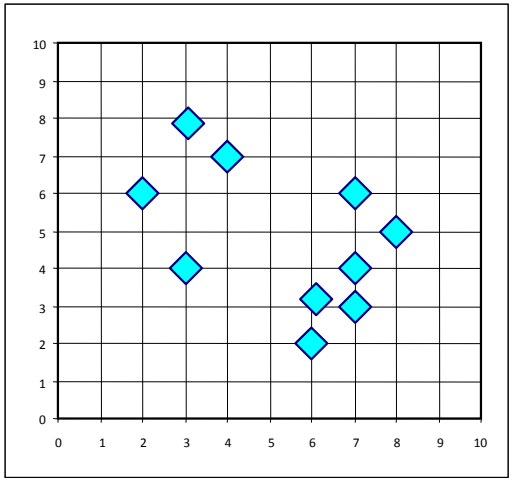
- Strength:
  - *Relatively efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
  - scalable
- Comment: Often terminates at a *local optimum*.
- Weakness
  - Applicable only when *mean* is defined. (what about categorical data?)
  - Need to specify *k,* the *number* of clusters, in advance
  - Run multiple times with different cluster centers
  - Sensitive to noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

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
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling

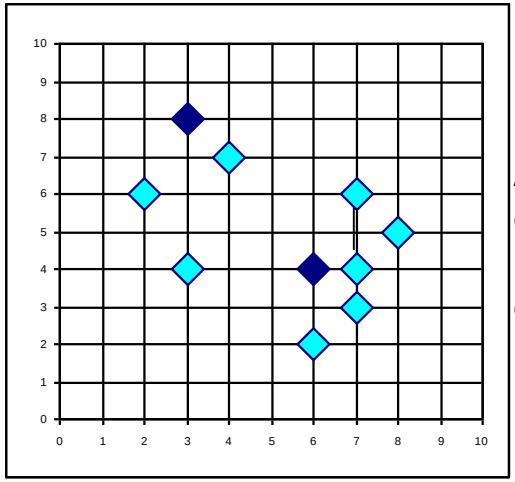
A Typical K-Medoids Algorithm (PAM)

Total Cost = 20

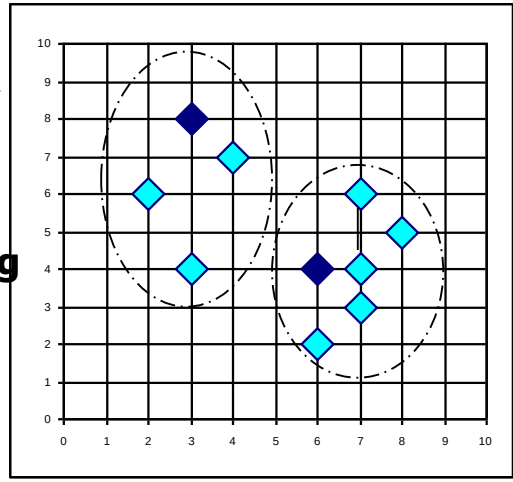


K=2

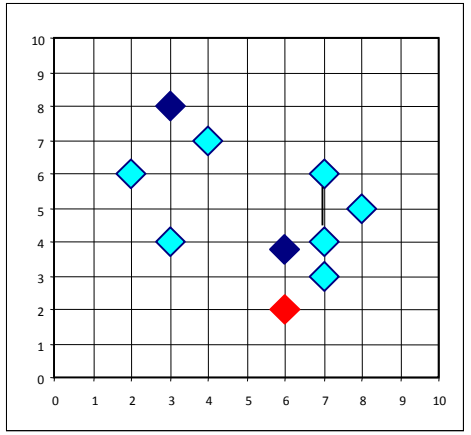
Arbitrary choose k object as initial medoids



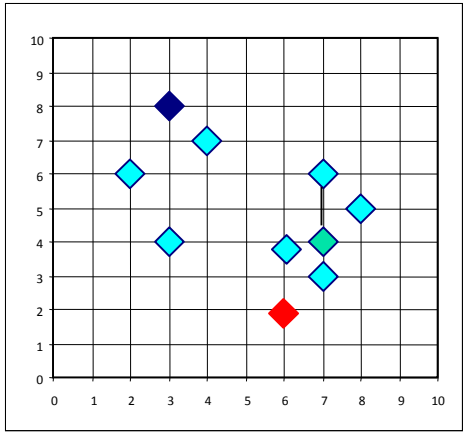
Assign each remaining object to nearest medoids



Randomly select a nonmedoid object,O_{ramdom}



Compute total cost of swapping



Swapping O and O_{ramdom}

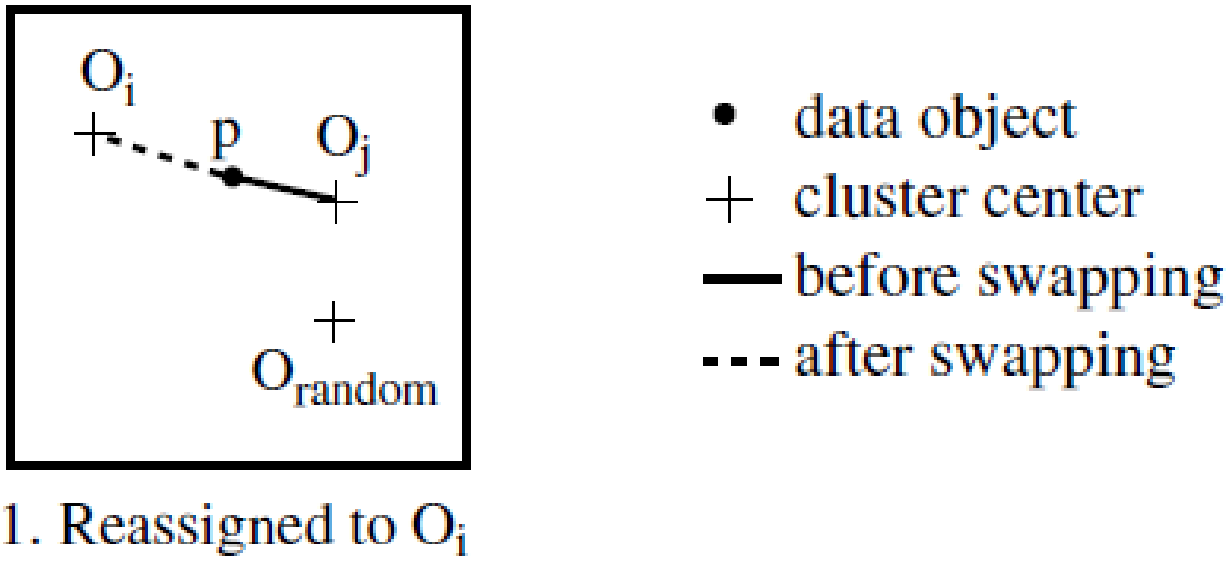
If quality is improved.

The partitioning method is performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point. That is, an **absolute-error criterion** is used, defined as

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j|,$$

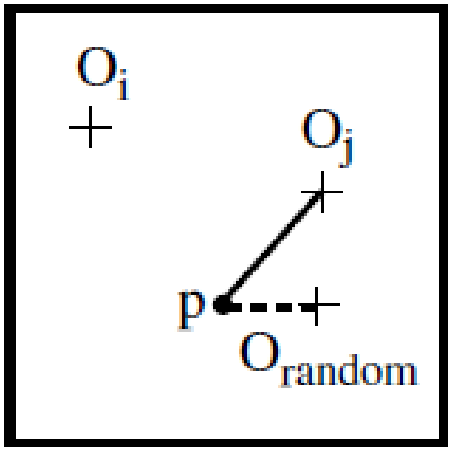
E is the sum of the absolute error for all objects in the data set;
p is the point in space representing a given object in cluster *Cj*;
oj is the representative object of *Cj*.

Case 1: ***p*** currently belongs to representative object, ***oj***. If ***oj*** is replaced by o_{random} as a representative object and ***p*** is closest to one of the other representative objects, ***oi***, $i \neq j$, then ***p*** is reassigned to ***oi***.



1. Reassigned to O_i

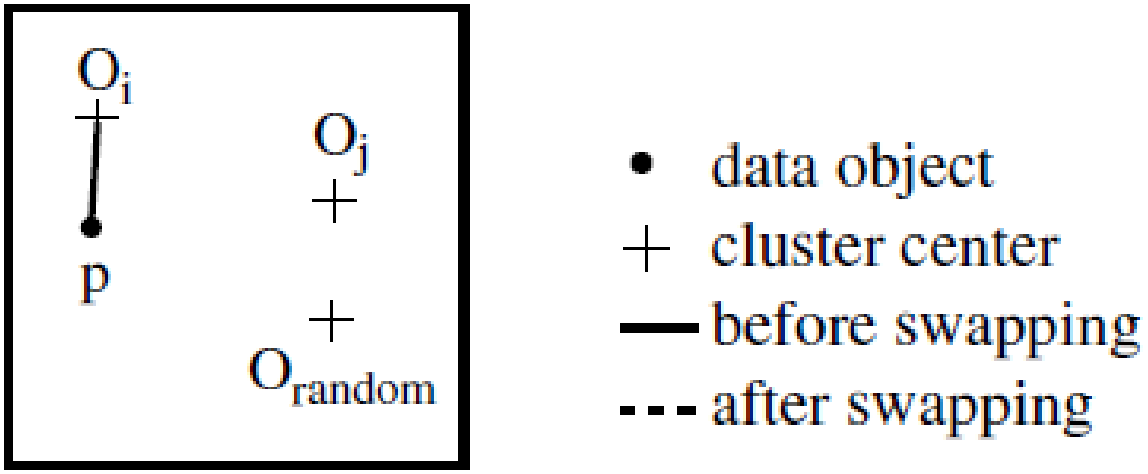
Case 2: ***p*** currently belongs to representative object, ***oj***. If ***oj*** is replaced by o_{random} as a representative object and ***p*** is closest to o_{random} , then ***p*** is reassigned to o_{random}



2. Reassigned to O_{random}

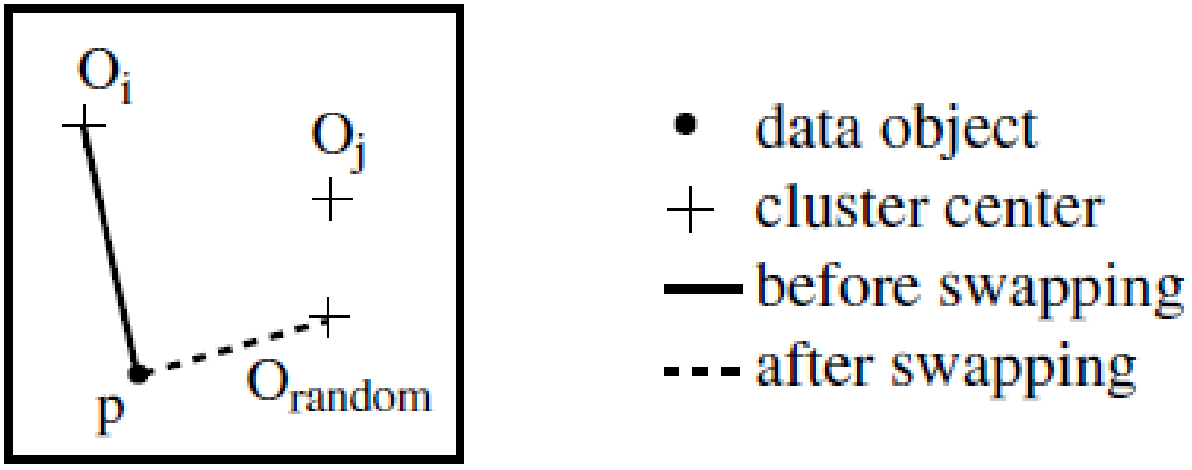
- data object
- cluster center
- before swapping
- after swapping

Case 3: ***p*** currently belongs to representative object, ***oi***, $i \neq j$. If ***oj*** is replaced by $\boldsymbol{o}_{random}$ as a representative object and ***p*** is still closest to *oi*, then the assignment does not change.



3. No change

Case 4: ***p*** currently belongs to representative object, ***oi***, $i \neq j$. If ***oj*** is replaced by $\boldsymbol{o}_{random}$ as a representative object and ***p*** is closest to $\boldsymbol{o}_{random}$, then ***p*** is reassigned to $\boldsymbol{o}_{random}$



4. Reassigned to O_{random}

Algorithm: *k*-medoids. PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

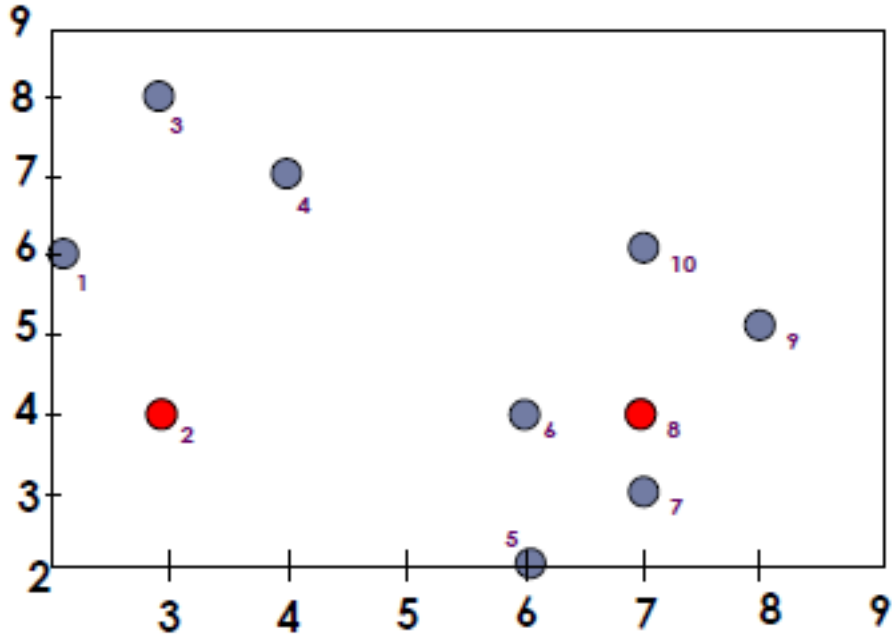
Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) repeat
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative object, o_j, with o_{random};
(6) if $S < 0$ then swap o_j with o_{random} to form the new set of *k* representative objects;
(7) until no change;

Example

- Cluster the following data set of ten objects into two clusters i.e. $k = 2$.

X_1	**2**	**6**
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6



Let us assume $c_1 = (3,4)$ and $c_2 = (7,4)$
So, c_1 and c_2 are selected as medoids.

Example

- Cluster the following data set of ten objects into two clusters i.e. $k = 2$.

X_1	**2**	**6**
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6

$$\text{cost}(x, c) = \sum_{i=1}^{d} |x_i - c_i|$$

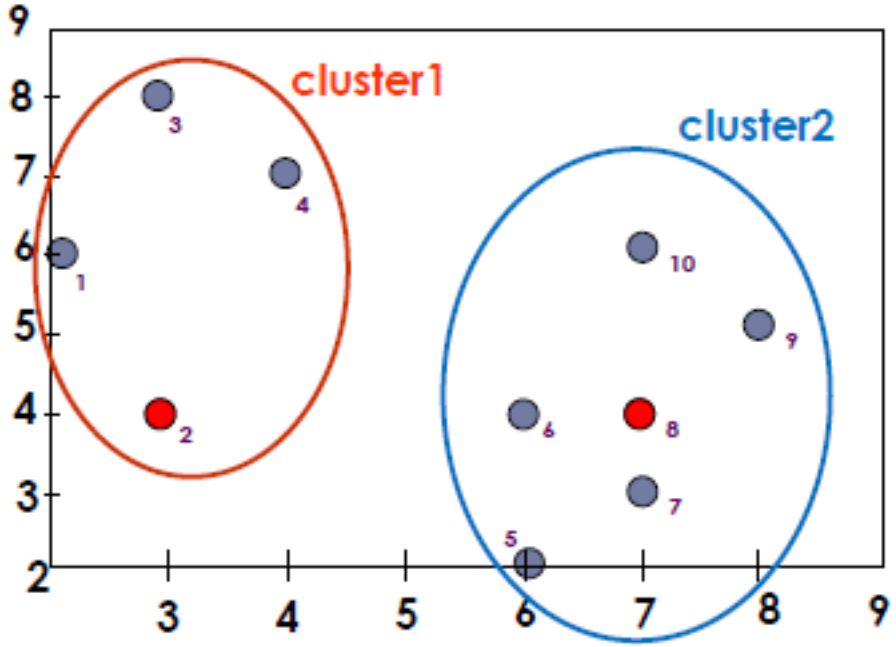
x is any data object, c is the medoid, and d is the dimension of the object

i		c_1	Data objects (X_i)		Cost (distance)
1	3	4	2	6	**3**
3	3	4	3	8	**4**
4	3	4	4	7	**4**
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	3	5
9	3	4	8	5	6
10	3	4	7	6	6

i		c_2	Data objects (X_i)		Cost (distance)
1	7	4	2	6	7
3	7	4	3	8	8
4	7	4	4	7	6
5	7	4	6	2	**3**
6	7	4	6	4	**1**
7	7	4	7	3	**1**
9	7	4	8	5	**2**
10	7	4	7	6	**2**

$Cluster_1 = \{(3,4)(2,6)(3,8)(4,7)\}$
$Cluster_2 = \{(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)\}$

total cost = 20.



Cluster1 = $\{O_1, O_2, O_3, O_4\}$

Cluster2 = $\{O_5, O_6, O_7, O_8, O_9, O_{10}\}$

$$\begin{aligned}\text{total cost} &= \{\text{cost}((3,4),(2,6)) + \text{cost}((3,4),(3,8)) + \text{cost}((3,4),(4,7))\} \\ &+ \{\text{cost}((7,4),(6,2)) + \text{cost}((7,4),(6,4)) + \text{cost}((7,4),(7,3)) \\ &+ \text{cost}((7,4),(8,5)) + \text{cost}((7,4),(7,6))\} \\ &= (3+4+4)+(3+1+1+2+2) \\ &= 20\end{aligned}$$

- **Select one of the nonmedoids O′**
- **Let us assume O′ = (7,3)**
- **So now the medoids are c_1(3,4) and O′(7,3)**
- **If c_1 and O′ are new medoids, calculate the total cost involved**

i		c_1	Data objects (X_i)		Cost (distance)
1	3	4	2	6	**3**
3	3	4	3	8	**4**
4	3	4	4	7	**4**
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	4	4
9	3	4	8	5	6
10	3	4	7	6	4

i		O′	Data objects (X_i)		Cost (distance)
1	7	3	2	6	8
3	7	3	3	8	9
4	7	3	4	7	7
5	7	3	6	2	**2**
6	7	3	6	4	**2**
7	7	3	7	4	**1**
9	7	3	8	5	**3**
10	7	3	7	6	**3**

$$\text{total cost} = 3 + 4 + 4 + 2 + 2 + 1 + 3 + 3$$
$$= 22$$

cost of swapping medoid from c_2 to O′ is more.

So moving to O′ would be a bad idea, the previous choice was good

What Is the Problem with 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 efficiently for small data sets but does not **scale well** for large data sets.
  - $O(k(n-k)^2)$ for each iteration

    where n is # of data, k is # of clusters
- Work for small data sets (100 objects in 5 clusters)
- Not efficient for medium and large data sets

➔Sampling based method,

CLARA(Clustering LARge Applications)

CLARA (Clustering Large Applications) (1990)

- *CLARA* (Kaufmann and Rousseeuw in 1990)

  Built in statistical analysis packages, such as S+
- It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output
- Strength: deals with larger data sets than *PAM*
- Weakness:
  - 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

CLARA Algorithm

Input: Database of D objects.

repeat for m times

draw a sample $S \subseteq D$ randomly from D.

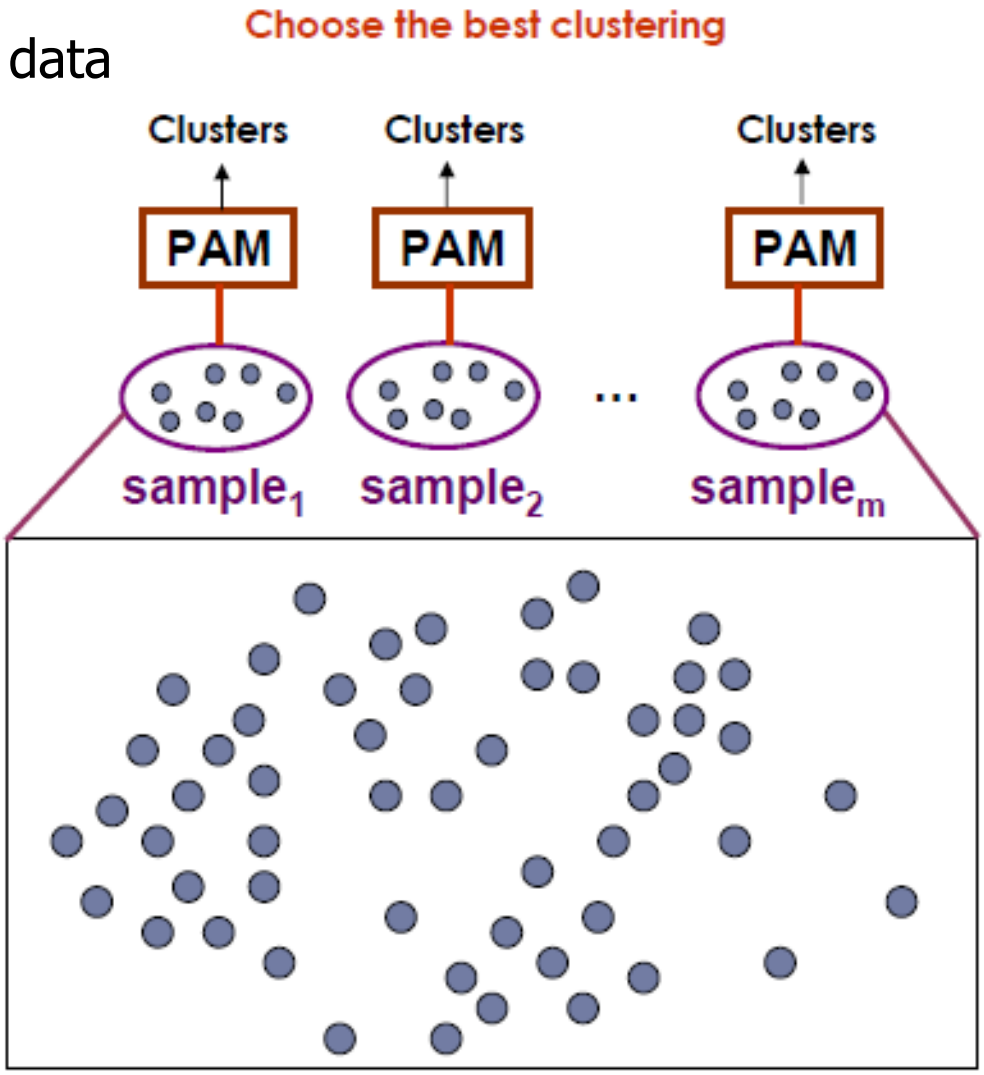
call PAM (S, k) to get k medoids.

classify the entire data set D to C_1, C_2 ... C_k.

calculate the quality of clustering as the average dissimilarity.

end.

- Draw multiple samples of the data set
- Apply PAM to each sample
- Return the best clustering



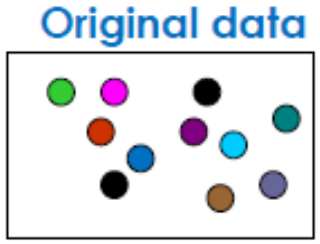
CLARANS ("Randomized" CLARA) *(1994)*

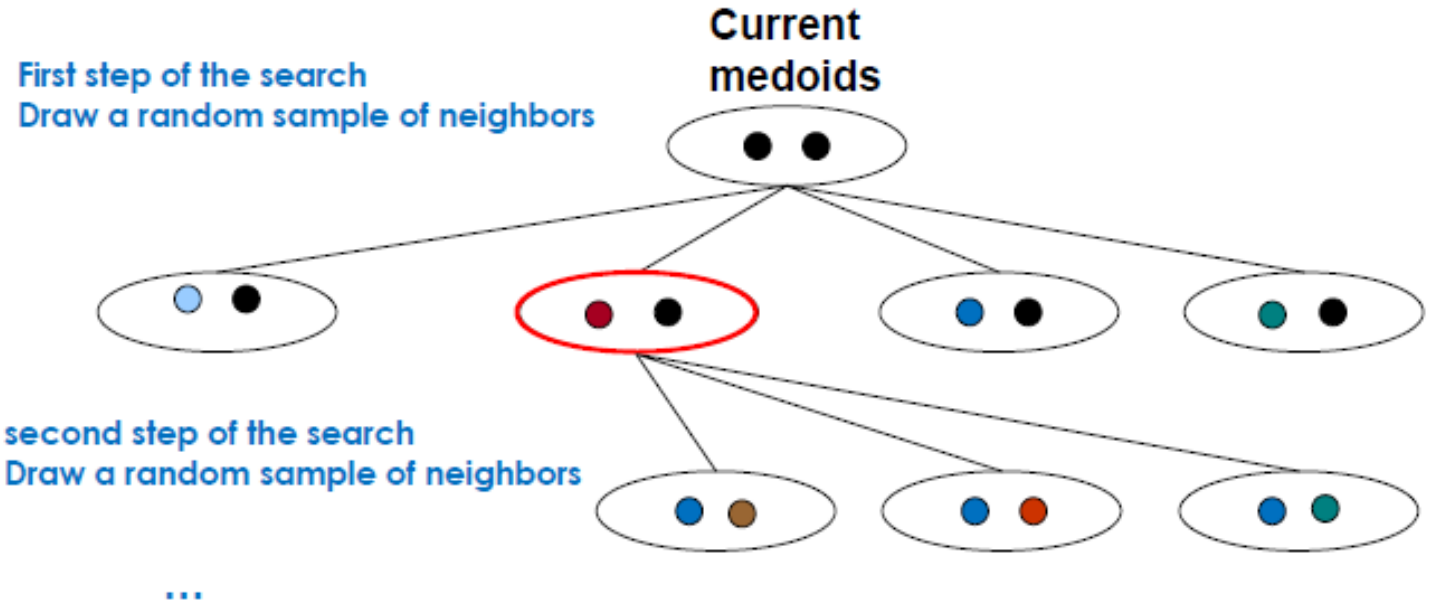
- *CLARANS* (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)
- CLARANS draws sample of neighbors dynamically
- The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of *k* medoids
- If the local optimum is found, *CLARANS* starts with new randomly selected node in search for a new local optimum
- It is more efficient and scalable than both *PAM* and *CLARA*

CLARANS: The idea

CLARANS

- Does not confine the search to a localized area
- Stops the search when a local minimum is found
- Finds several local optimums and output the clustering with the best local optimum





The number of neighbors sampled from the original data is specified by the user

CLARANS Algorithm

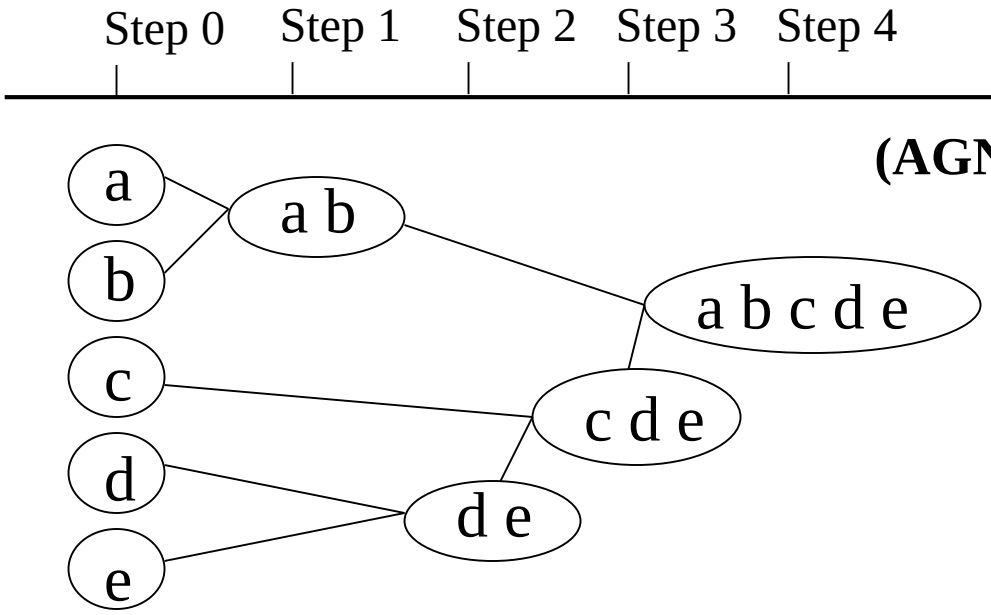
Input (D, k, maxneighbor and numlocal)
select arbitrarily k representative objects.
mark these objects as "selected" and all other objects as non-selected. Call it current.
set $e = 1$.
do while ($e \leq$ numlocal)
    set $j = 1$
    do while ($m \leq$ maxneighbor)
        select randomly a pair (j, h) such that O_j is a selected object and O_h is a non-selected object.
        compute the cost C_{jh}.
        if C_{jh} is negative
            "update current"
            mark O_j non-selected, O_h selected and $m = 1$
        else
            increment $m \leftarrow m + 1$
    end do
    compare the cost of clustering with "mincost"
    if current_cost < mincost
        mincost ← current_cost
        best_node ← current
    increment $e \leftarrow e + 1$
end do
return "best node"

Numlocal: no. of optimal medoid sets
Maxneighbor: no. of pairs for swapping

Hierarchical Clustering

- Hierarchical decomposition of the data base.
- A hierarchical clustering method works by grouping data objects into a hierarchy or tree of clusters.
- Use: summarize and represent the data in a compressed way
- This method needs a termination condition – desired number of clusters, iterations etc.

Hierarchical Clustering



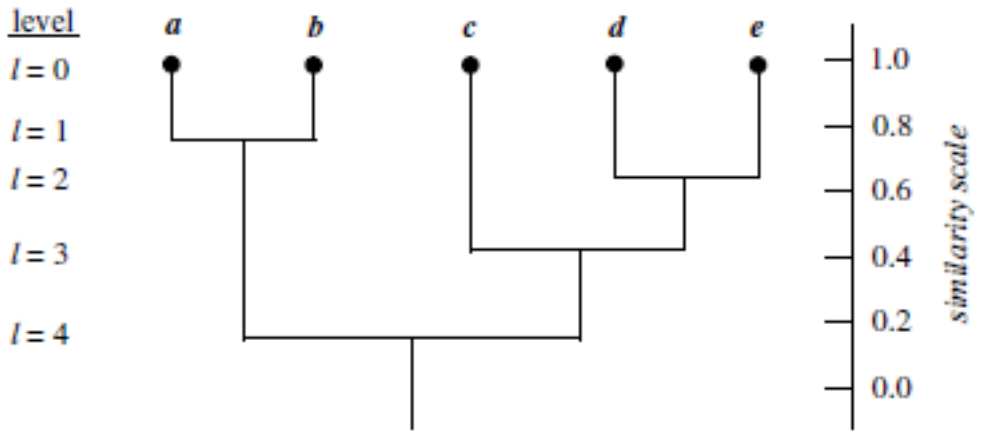
Agglomerative

(AGNES - (AGglomerative NESting**)**

Divisive

(DIANA - DIvisive ANAlysis**)**

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits



Dendrogram representation for hierarchical clustering of data objects $\{a, b, c, d, e\}$.

Challenge with divisive method

- how to partition large cluster into several smaller ones? 2^{n-1}-1 possible ways! To partition a set of n objects into two exclusive subsets.
- Uses heuristics – may lead to inaccurate results

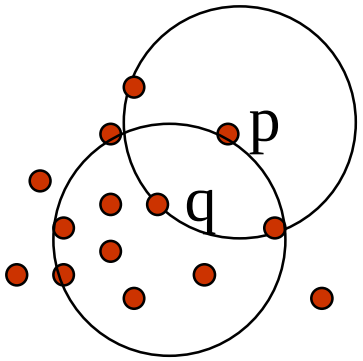
Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Density of an object "O" can be measured by the number of objects close to it
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition

DBSCAN: Density Based Spatial Clustering of Applications with Noise :Basic Concepts

- Є – neighbourhood of an object
  - $N_Є(O_i)$: $\{O_j$ *belongs to* $D \mid dist(O_i, O_j) <= Є$ (radius)$\}$
- *MinPts*: Minimum number of points in an Є -neighbourhood of that point
- *Core object*
  - $\mid N_Є(O_i) \mid >=$ MinPts
- Directly density-reachable A point *p* is directly density-reachable from a point *q* w.r.t. Є, *MinPts* if
  - *p* belongs to $N_Є(q)$
  - core point condition:

$$|N_Є(q)| >= MinPts$$



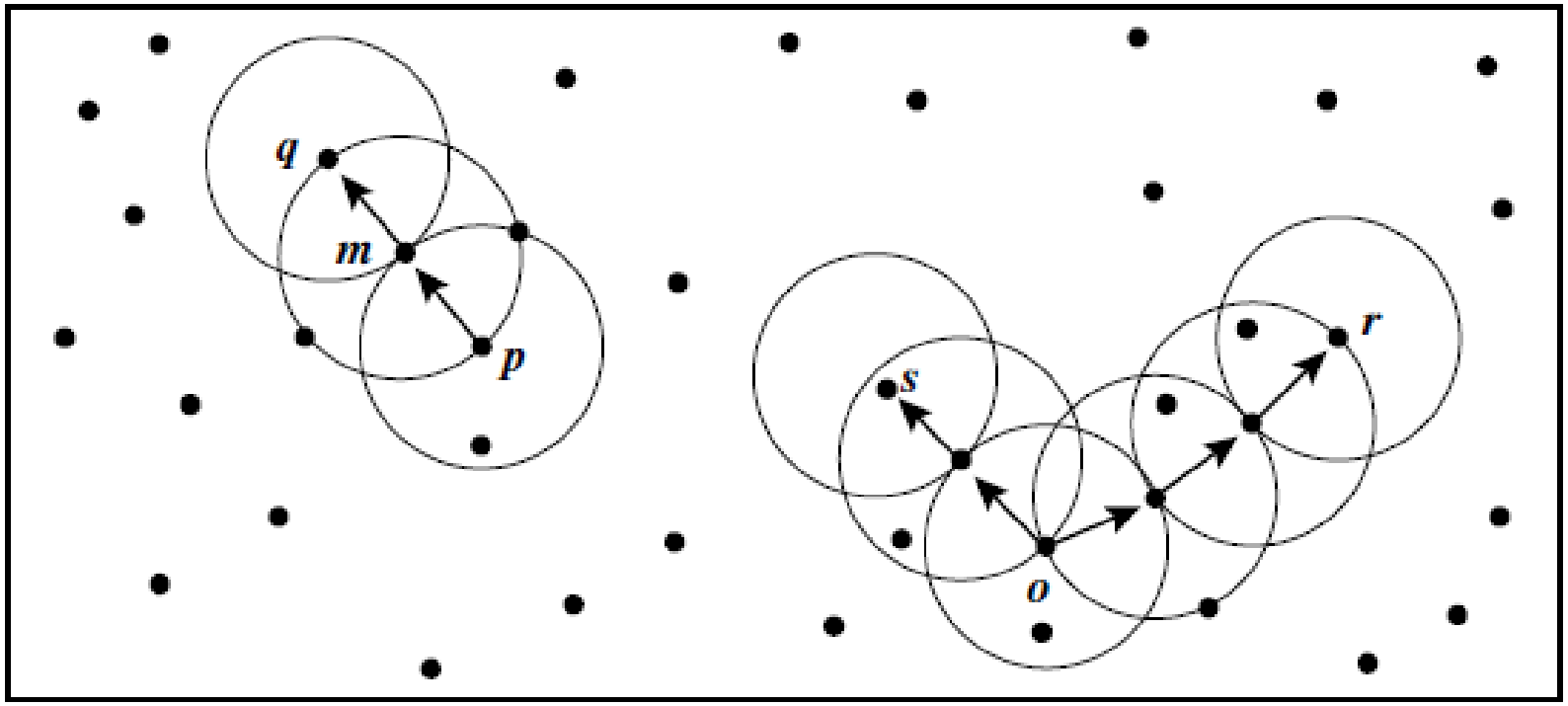
MinPts = 5

Є = 1 cm

Density-Reachable and Density-Connected

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

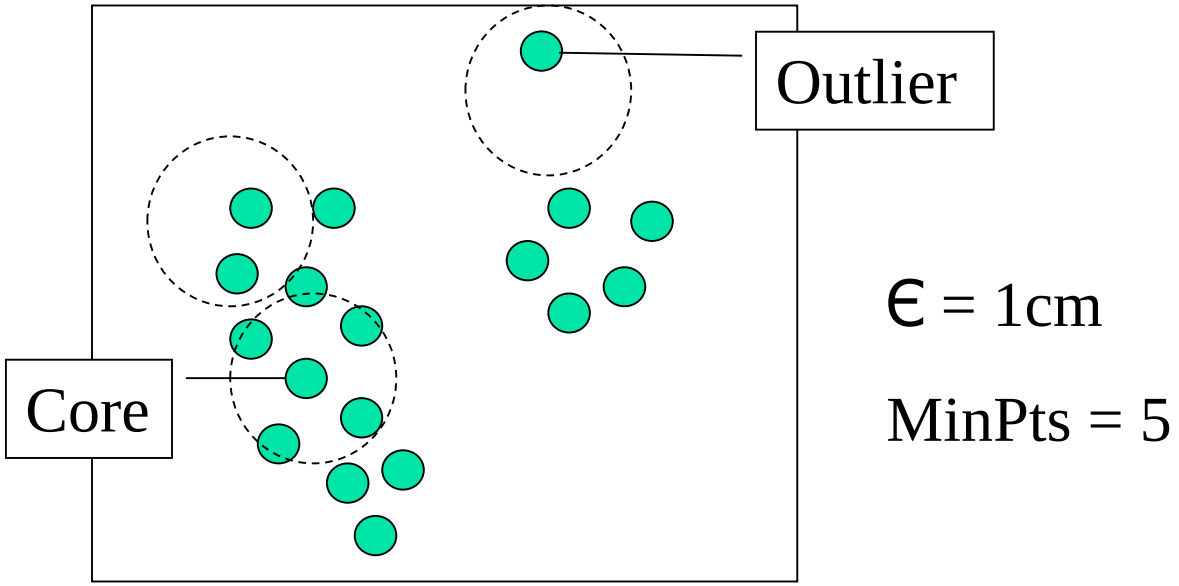
Minpts=3



- Core objects: m, p, o, r
- Directly Density reachable: q from m, m from p, p from m
- Density reachable: r and s from o, o from r
- Density connected: o,r,s
- Note: p is not density reachable from q

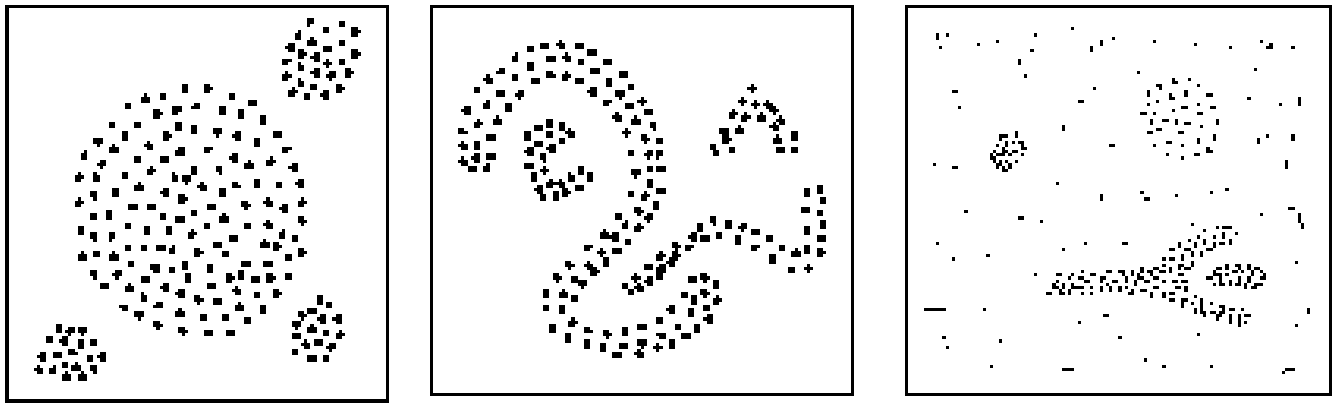
- Cluster (C): non empty subset of DB w.r.t. Є and MinPts
  - For all O_i, O_j belongs to DB, if O_i belongs to C and O_j is density-reachable from O_i w.r.t. Є and MinPts, then O_j belongs to C
  - For all O_i, O_j belongs to C, O_i is density-connected to O_j w.r.t. Є and MinPts
- Noise : the set of data objects in DB which do not belong to any cluster C_i

- Core object
- Non core object



- Advantages
  - DBSCAN does not require to specify the number of clusters.
  - DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster.
  - DBSCAN has a notion of noise.
  - DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database.
- Disadvantages
  - The quality of DBSCAN depends on the distance measure used
  - DBSCAN cannot cluster data sets well with large differences in densities, since the minPts- ϵ combination cannot be chosen appropriately for all clusters.

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 Algorithm

Algorithm DBSCAN (D, ε, MinPts)
Input: Database of objects D
do for all $O \in D$
    if O is unclassified
        call function **expand_cluster**(O, D, ε, MinPts)
end do.

Function expand_cluster (O, D, ε, MinPts):
get the ε-neighbourhood of O as $N_\varepsilon(O)$
    if $|N_\varepsilon(O)| <$ MinPts,
        mark O as noise
        return
    else
        select a new cluster_id and mark all objects of $N_\varepsilon(O)$ with this cluster-id and put them into candidate-objects.
        do while candidate-objects is not empty
            select an object from candidate-objects as current_object
            delete current-object from candidate-objects
            retrieve N_ε(current-object)
            if $|N_\varepsilon$(current-object)$| \geq$ MinPts
                select all objects in N_ε(current-object) not yet classified or marked as noise,
                mark all of the objects with cluster_id,
                include the unclassified objects into candidate-objects
        end do
return.