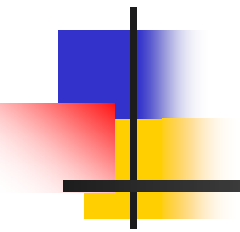
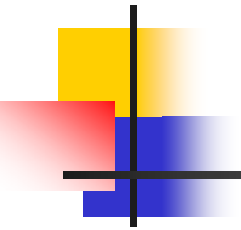


Clustering Analysis

Part One



Clustering

- What is cluster analysis?
- Types of data in cluster analysis
- A categorization of major clustering methods
  - Partitioning methods
  - Hierarchical methods
  - Model-based clustering methods
- Outlier analysis
- Summary

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.
  - Grouping a set of data objects into clusters, such that objects within each cluster are similar to each other, objects in different clusters are dissimilar to each other.
- Clustering is unsupervised classification:

  Objects are not labeled with predefined classes.
- Typical applications.
  - As a stand-alone tool to get insight into data distribution.
  - As a preprocessing step for other algorithms.

General Applications

- Pattern recognition
- (Spatial) data analysis
- Image processing
- Economic science (especially market research)
- WWW
  - Automatic document categorization
  - Web usage mining: cluster web log data to discover groups of similar access patterns
- Business : customer groups
- Biology: animal and plant taxonomy
  Categorize genes by functionality
- And many more ...

What Is Good Clustering?

- A good clustering method will produce high quality clusters with.
  - High intra-class similarity.
  - Low inter-class similarity.
- The quality of a clustering result depends on both the similarity measure used by the method and its clustering approach used.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden 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
- Insensitive to order of input records
- High dimensionality
- Interpretability and usability

Data Structures

- Data matrix

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

- Dissimilarity matrix

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

Measure the Quality of Clustering

- Dissimilarity/similarity metric: similarity is expressed in terms of a distance function, which is typically metric:
  d(i, j).
- There is a separate “quality” function that measures the “goodness” of a cluster.
- The definitions of distance functions are usually very different for interval-scaled, boolean, categorical, ordinal variables, and temporal data.
- 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

Type of Data in Clustering Analysis

- Interval-scaled variables
- Binary variables
- Nominal, and ordinal variables
- Variables of mixed types:
- Temporal

Interval-valued Variables

- Standardize data
  - Calculate the mean absolute deviation:

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

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

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

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

- Using mean absolute deviation is more robust than using standard deviation

Similarity and Dissimilarity Between Objects

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

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

Where $i = (x_{i1}, x_{i2}, ..., x_{ip})$ and $j = (x_{j1}, x_{j2}, ..., x_{jp})$ are two p-dimensional data objects, and q is a positive integer

- If $q = 1$, d is Manhattan distance

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{ip} - x_{jp}|$$

If $q = 2$, d is Euclidean distance:

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2)}$$

- Properties
  - $d(i,j) \geq 0$
  - $d(i,i) = 0$
  - $d(i,j) = d(j,i)$
  - $d(i,j) \leq d(i,k) + d(k,j)$

Triangular inequality

Binary Attributes

- A contingency table for binary data

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

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

- Jaccard coefficient (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

Jaccard coefficient

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

$$d(jack, jim) = \frac{1 + 1}{1 + 1 + 1} = 0.67$$

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

Nominal Attributes

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

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

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

Ordinal Attributes

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

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

  - compute the dissimilarity using methods for interval-scaled attributes

Attributes of Mixed Types

- A database may contain different types of attributes
  - symmetric binary, asymmetric binary, nominal, ordinal, and interval.
- One may use a weighted formula to combine their effects.

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

  - *f* is binary or nominal:

    $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{(f)} = 1$ o.w.
  - *f* is interval-based: use the normalized distance
  - *f* is ordinal
    - compute ranks r_{if} and $z_{if} = \frac{r_{if} - 1}{M_f - 1}$
    - and treat z_{if} as interval-scaled

Major Clustering Approaches

- 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
- Model-based: A model is hypothesized 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: 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* (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

The *K-Means* Clustering Method

- **Objective**: to form a set of clusters that are as compact and separated as possible
- **Distance Measure**: Euclidean distance between data object and cluster center
- **Clustering criterion function**:

  mean squared error

$$E = \sum_{i=1}^{k} \sum_{p \subseteq C_i} | p - m_i |^2$$

p: a data object
C_i: cluster i
m_i: center of cluster i
k: number of clusters

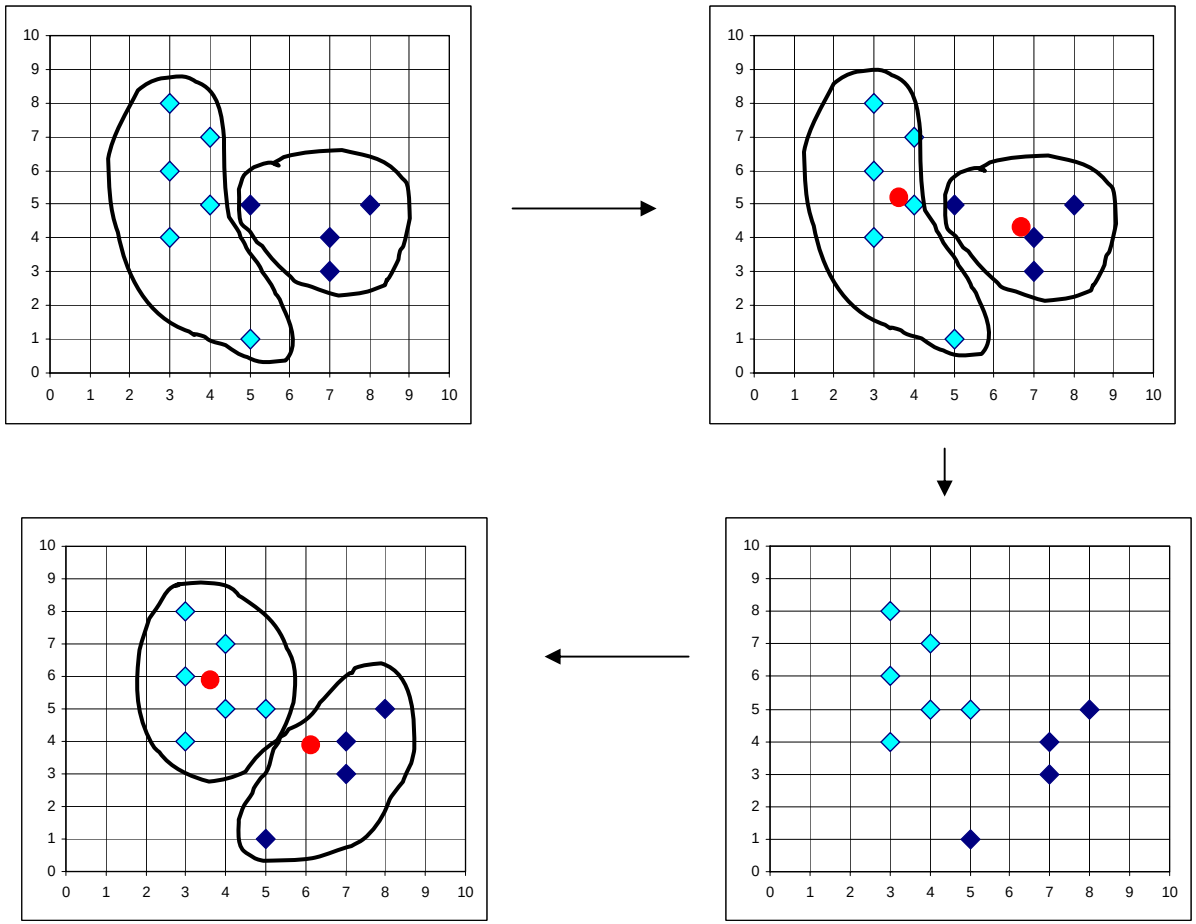
The *K-Means* Clustering Method

- **Approach**: Given *k*, the *k-means* algorithm is implemented as the following:
  - arbitrarily choose *K* objects as the initial cluster centers.
  - Repeat:
    - **Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.**
    - **Assign each object to the cluster with the nearest seed point.**

  stop when no more new assignment, or when clustering criterion function (mean squared error) converges.

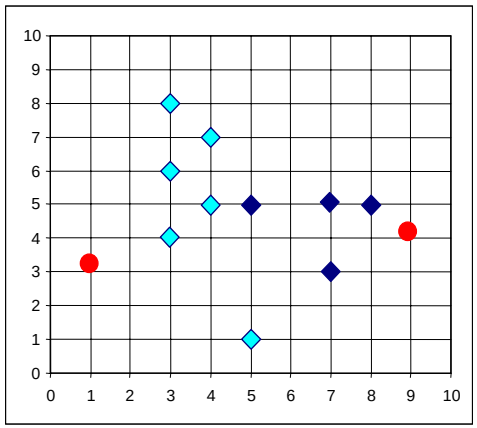
The *K-Means* Clustering Method

- Example



- Clusters are Represented by the Centers of the clusters
- center of a cluster may not correspond to any object

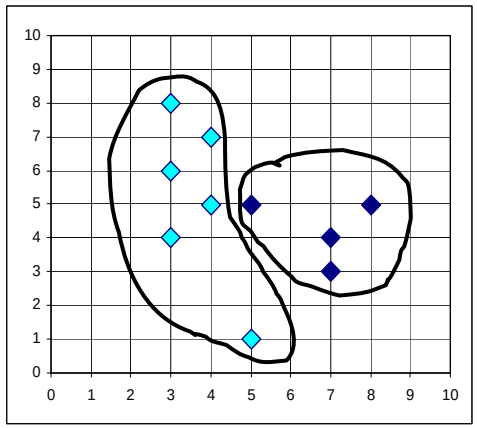
K-Means



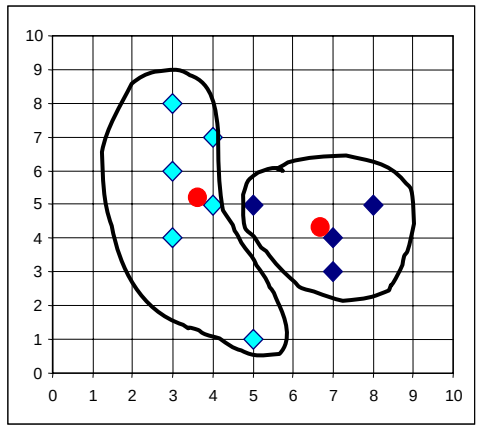
K=2

Arbitrarily choose K object as initial cluster center

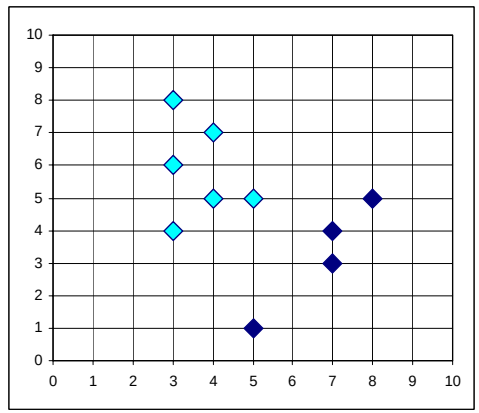
Assign each objects to most similar center



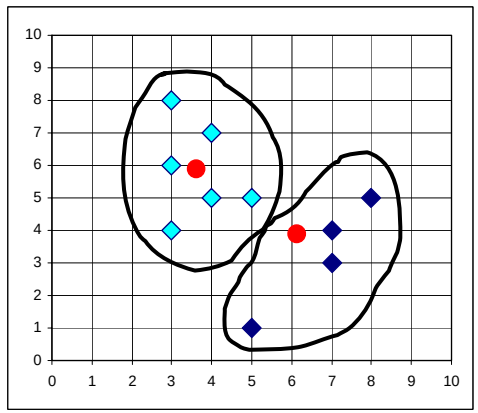
Update the cluster means



reassign



Update the cluster means

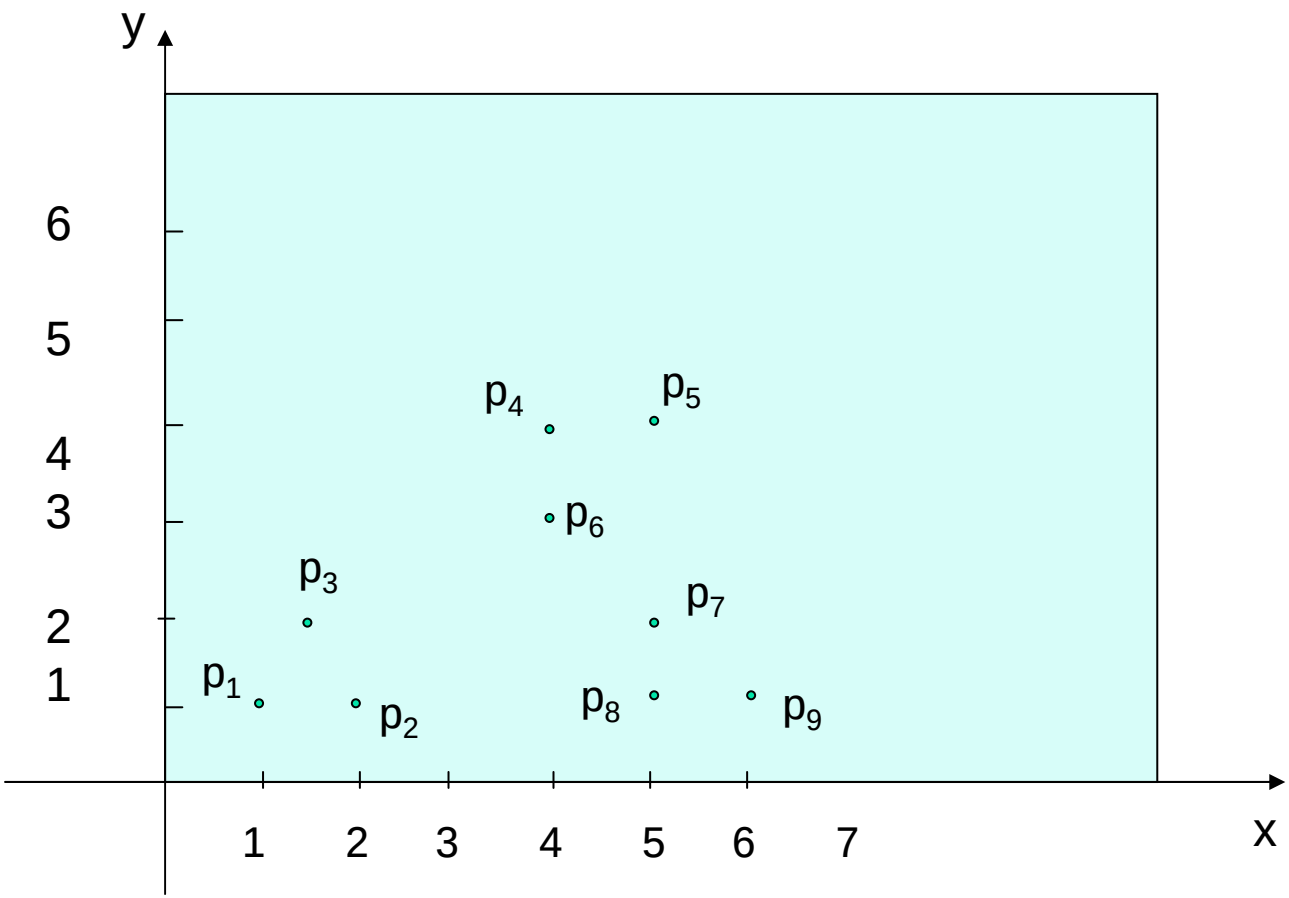


reassign

Example

Apply K-means clustering algorithm to partition the following data with 9 data objects:

P1(1, 1)
P2(2,1)
P3(1.5,2)
P4(4, 4)
P5(5,4)
P6(4,3)
P7(5,2)
P8(5,1)
P9(6,1)



Comments on the *K-Means* Method

- Strength
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.
  - 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
  - Sensitive to initial seed selection
  - Unable to handle noisy data and *outliers*

Variations of the *K-Means* Method

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes* (Huang'98)
  - 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, 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

PAM (Partitioning Around Medoids)

- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- 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

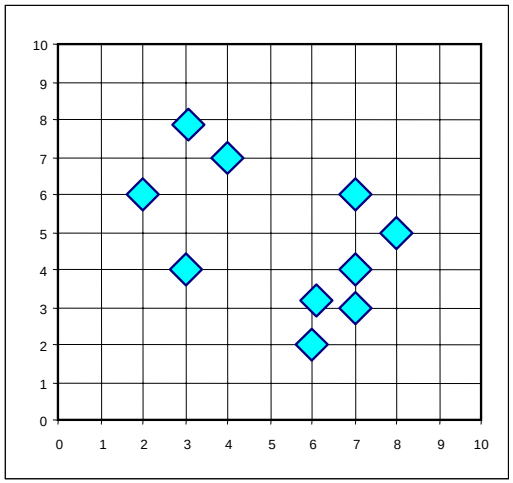
K-Medoids

- Arbitrarily choose K objects as the initial medoids;
- Repeat:
  - Assign each remaining object to the cluster with the nearest medoids;
  - Randomly select a nonmedoid object O_{random};
  - Compute the total cost, S, of swapping O_j with O_{random};
  - If S<0, then swap O_j with O_{random} to form the new set of k medoids;
- Until no change;

k-Medoids

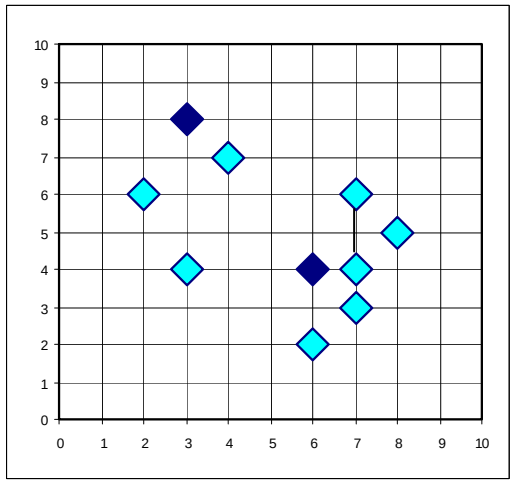
Total swapping cost $TC_{ih}=\sum_p C_{pih}$

Total Cost = 20

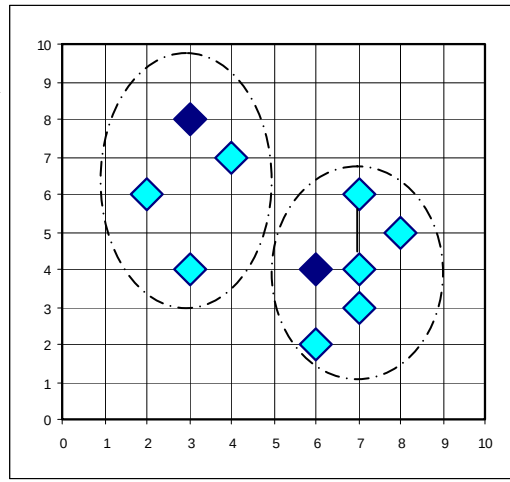


K=2

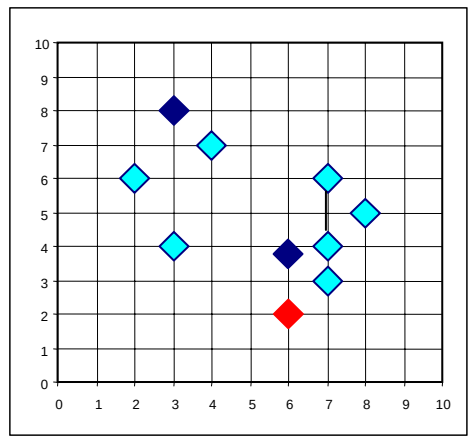
Arbitrary choose k object as initial medoids



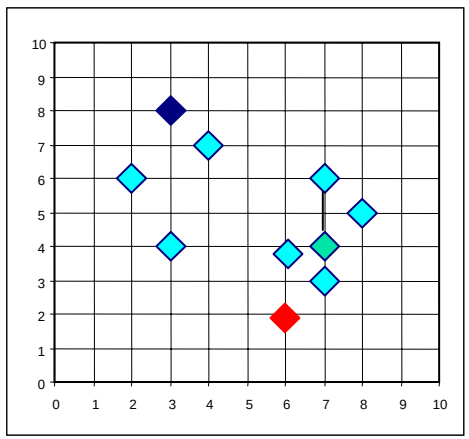
Assign each remaining object to nearest medoids



Randomly select a nonmedoid object,O_{ramdom}



Compute total cost of swapping



Total Cost = 26

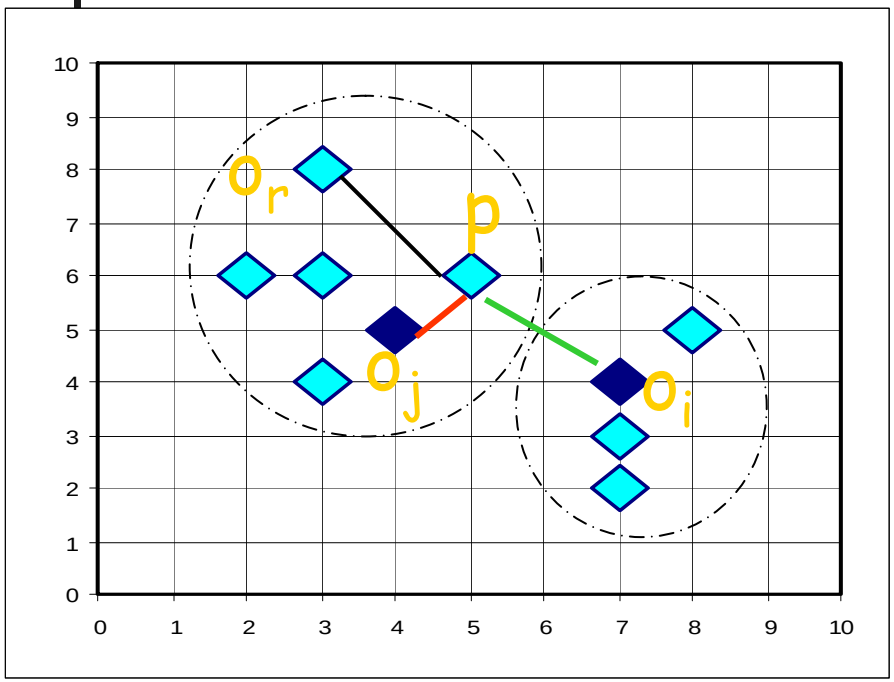
Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

PAM Clustering



Replace o_j with o_r

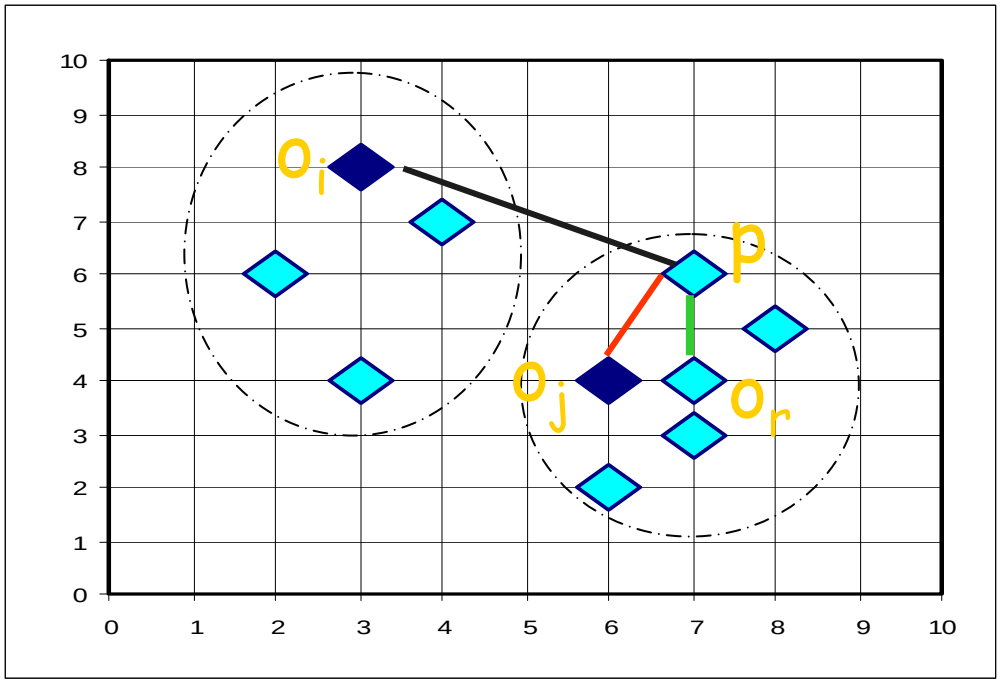
$p \in o_j$

p is now closer to o_i $i \neq j$

Reassign p to O_i

$$C_{p,j,r} = d(p, o_i) - d(p, o_j)$$

PAM Clustering



Replace o_j with o_r

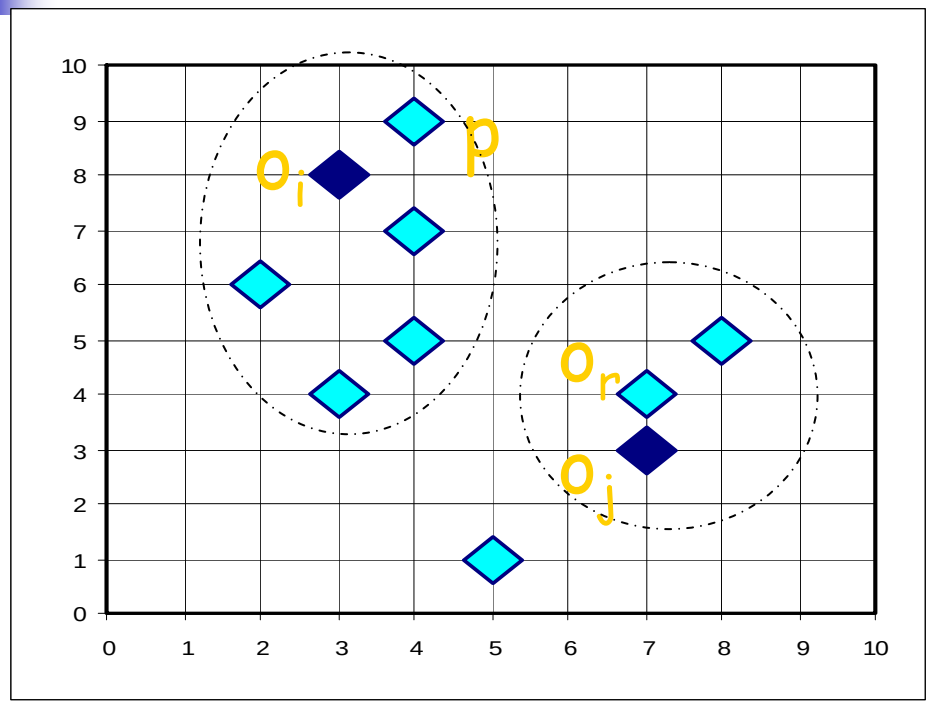
$p \in o_j$

p closest to o_r

Reassign p to O_r

$$Cp,j,r = d(p, o_r) - d(p, o_j)$$

PAM Clustering



Replace o_j with o_r

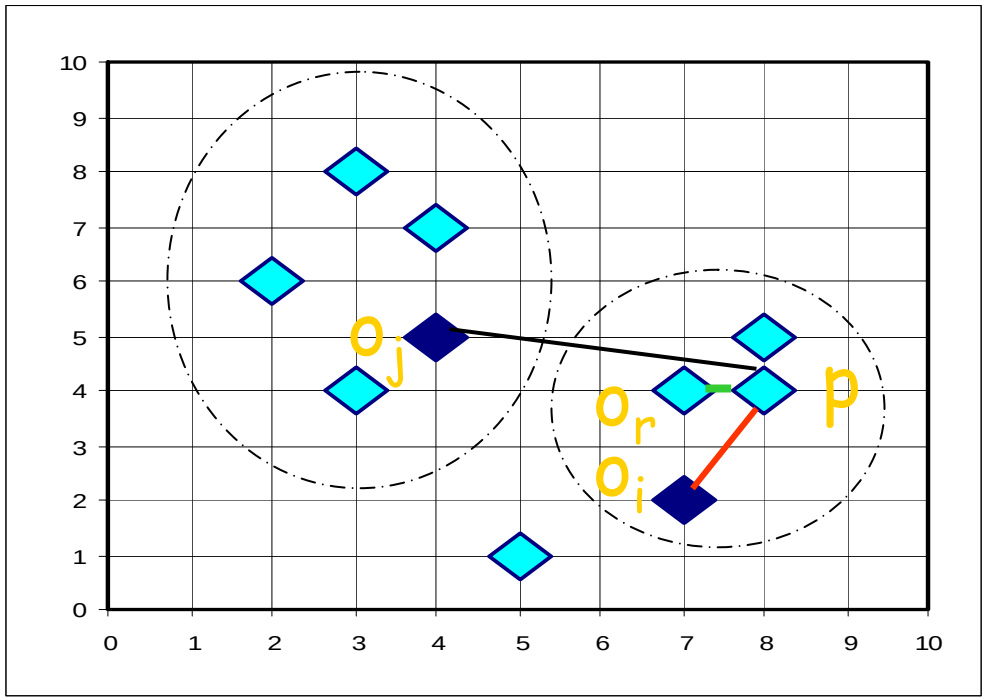
$p \in o_i$; $i \neq j$;

p still closest to o_i

no change

$$C_{p,j,r} = 0$$

PAM Clustering



Replace o_j with o_r

$p \in o_i$; $i \neq j$;

p closest to o_r

Reassign p to O_r

$$Cp,j,r = d(p, o_r) - d(p, o_i)$$

PAM Complexity Analysis

- Total k*(n-k) pairs of (O_i, O_h)
- For each pair of (O_i, O_h):
  - compute Tc_{ih} require the examination of (n-k) non-selected objects.
- Total complexity:

  $O(k*(n-k)^2)$

Compare K-means and PAM

- K-means is computationally more efficient
- K-means only handles numeric data
- PAM can handle different types of data
- PAM is better in terms of handling outliers in data

The CLARA algorithm

- Objective: to improve the computational efficiency of PAM, through sampling
- Basic idea:
  - draw a sample of the original data set, applies PAM on the sample, and finds the medoids of the sample.
  - Repeat the process a fixed number of times and return the medoids that generate the lowest average dissimilarity from the data objects
- Complexity: $O(k*(40+k)^2 + k*(n-k))$

The 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 the *k* medoids of the sample
- For each object O_j in the entire data set, determine which of the *k* medoids is the most similar to O_j.
- Calculate the average dissimilarity of the clustering obtained in the previous step. If this value is < current minimum, set current minimum to this value, and retain the current set of *k* medoids
- Return to step 1 to start the next iteration

CLARANS (“Randomized” CLARA)

- *CLARANS* (A Clustering Algorithm based on Randomized Search)
- 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*

The CLARANS Algorithm

1. Input *numlocal* and *maxneighbor*
   i=1, mincost=FLT_MAX, bestnode=NULL
2. *current* = an arbitrary *k* modiods
3. j=1
4. Pick random neighbor *S* of *current*, compute the cost difference between *S* and *current*
5. If S has lower cost, set *current* = S, goto 3
   else
   j=j+1;
   if (j <= *maxneighbor*) goto 4
   else
   if (cost(*current*) < *mincost*)
   mincost = cost(*current*)
   bestnode = *current*
6. i= i+1;
7. If (i <= *numlocal*)
   goto step 2
   else
   output *bestnode* and halt