AI 2002
Artificial Intelligence

Dr. Hashim Yasin

Unsupervised Learning

Unsupervised Learning

- In **unsupervised learning**, the agent learns patterns in the input even though **no explicit feedback** is supplied.
- **Unsupervised learning** occurs when no classifications are given and the *learner must discover categories and regularities in the data*.
- The most general example of unsupervised learning task is **clustering**:
  - potentially useful clusters developed from the input examples.
- For example, **a taxi agent** might gradually develop a concept of “good traffic days” and “bad traffic days”.

Clustering

K-means Clustering

- K-means is a partitioning clustering algorithm
- Let the set of data points (or instances) *D* be

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\},$$

  where
  - $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and
  - r is the number of attributes (dimensions) in the data.

- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means Clustering

- Basic Algorithm:

Select K points as the initial centroids.
repeat
    Form K clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change

Stopping/Convergence Criterion

1. No (or minimum) re-assignments of data points to different clusters,
2. No (or minimum) change of centroids, or
3. Minimum decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$$

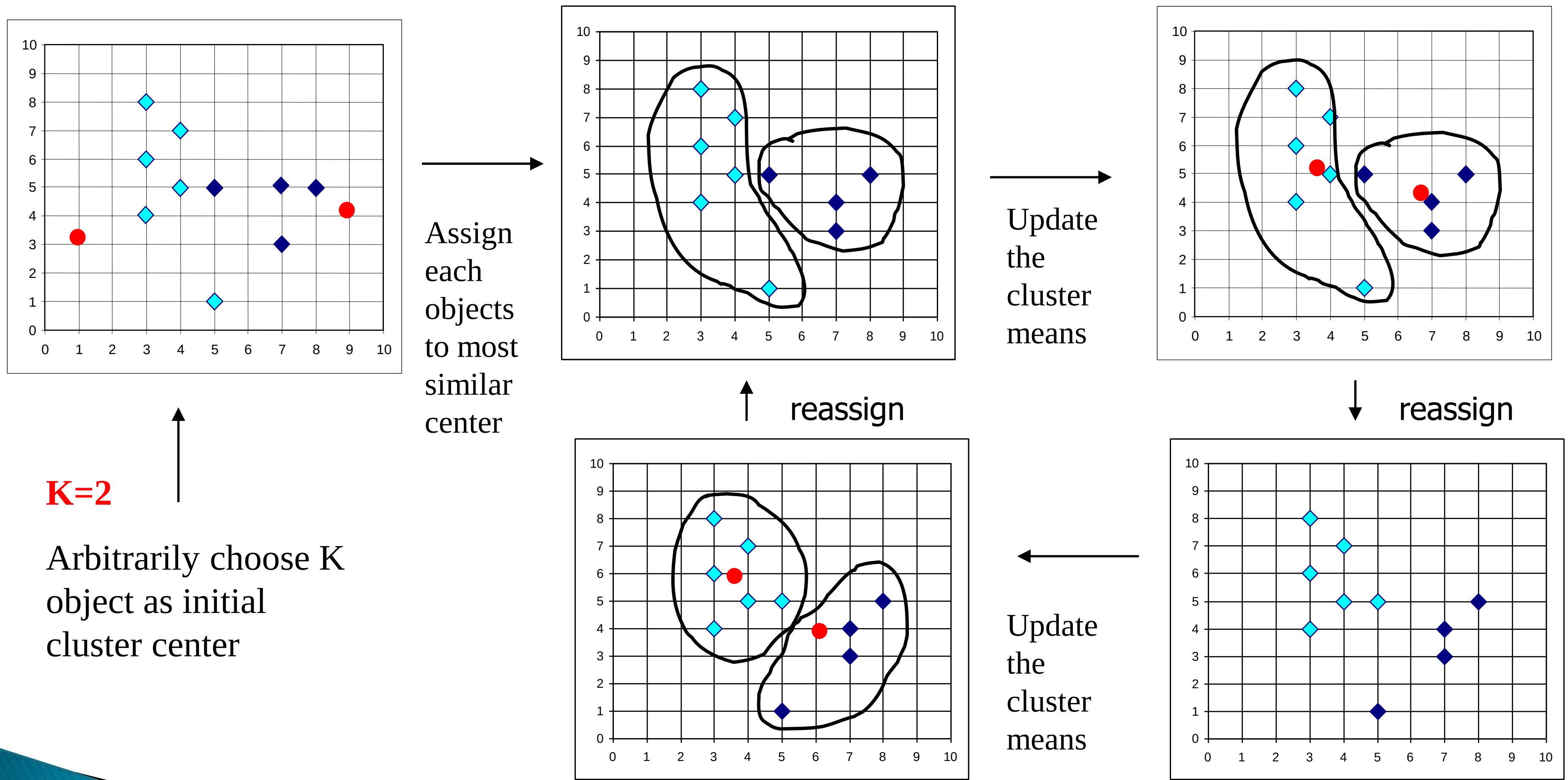
- C_j is the j^{th} cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

K-means Clustering--- Details

- *Initial centroids are often chosen randomly*.
  - Clusters produced vary from one run to another.
- The **centroid** is (typically) the mean of the points in the cluster.
- **‘Closeness’** is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.

- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’

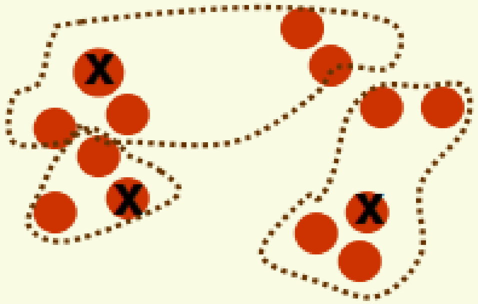
K-means Clustering Example



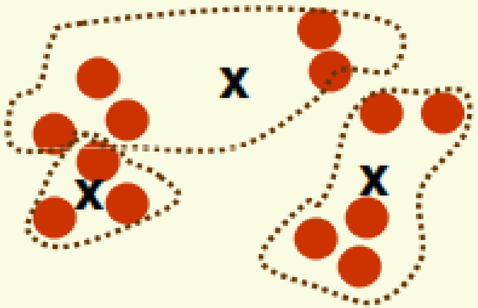
K-means Clustering

k = 3

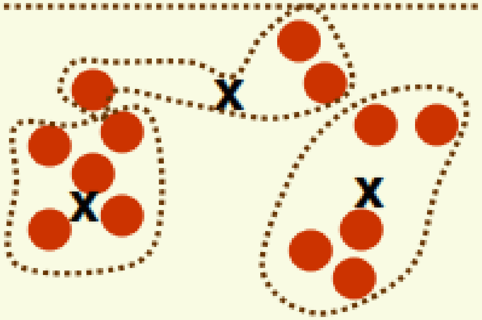
1. Initialize
   - pick ***k*** cluster centers arbitrary
   - assign each example to closest center



2. compute sample means for each cluster



3. reassign all samples to the closest mean



4. if clusters changed at step 3, go to step 2

K-means Clustering

- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - **Eliminate small clusters** that may represent outliers
  - **Split 'loose' clusters**, i.e., clusters with relatively high SSE
  - **Merge clusters that are 'close'** and that have relatively low SSE

Distance Function

- Most commonly used functions are
  - Euclidean distance and
  - Manhattan (city block) distance
- We denote distance with: *dist*($\mathbf{x}_i$, $\mathbf{x}_j$), where $\mathbf{x}_i$ and $\mathbf{x}_j$ are data points (vectors)
- They are special cases of Minkowski distance. q is positive integer.

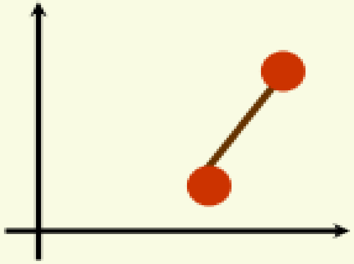
$$d(i,j) = \sqrt[q]{\underbrace{|x_{i1} - x_{j1}|^q}_{1^{st}\text{ dimension}} + \underbrace{|x_{i2} - x_{j2}|^q}_{2^{nd}\text{ dimension}} + \ldots + \underbrace{|x_{ip} - x_{jp}|^q}_{p^{th}\text{ dimension}}}$$

Distance (dissimilarity) Measures

Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

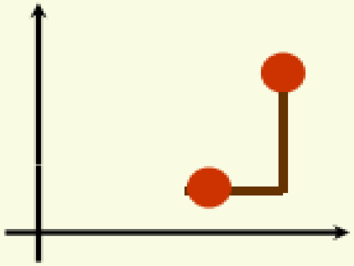
- translation invariant



Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

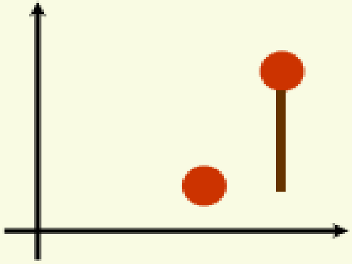
- approximation to Euclidean distance, cheaper to compute



Chebyshev distance

$$d(x_i, x_j) = \max_{1 \le k \le d} | x_i^{(k)} - x_j^{(k)} |$$

- approximation to Euclidean distance, cheapest to compute



K-means Clustering

- Time complexity for K-means clustering is

$$O(\, n \times K \times I \times d \,)$$

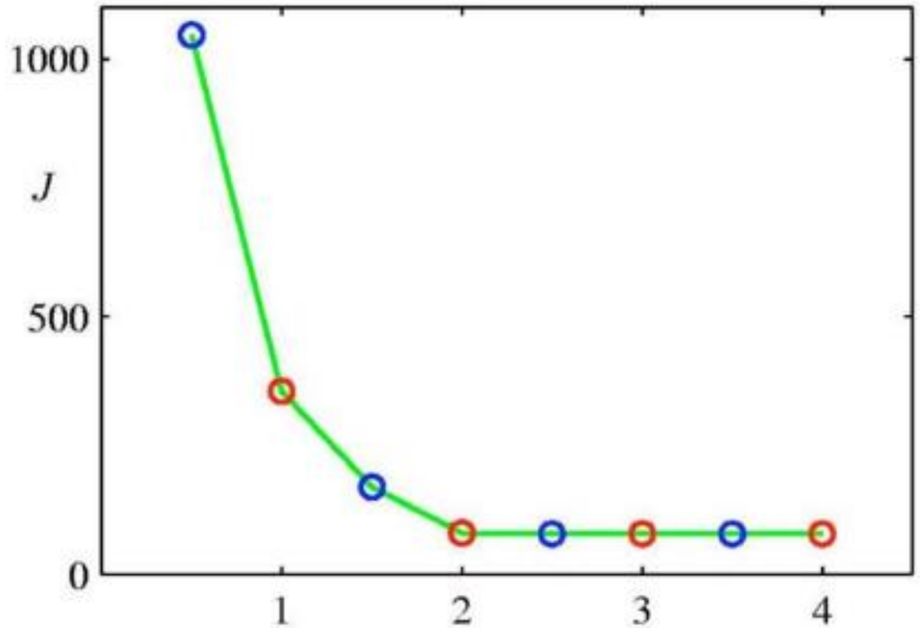
  - n = number of points,
  - K = number of clusters,
  - I = number of iterations,
  - d = number of attributes
- The storage required is

$$O((n + K)d)$$

  - n = number of points,
  - K = number of clusters,
  - d = number of attributes

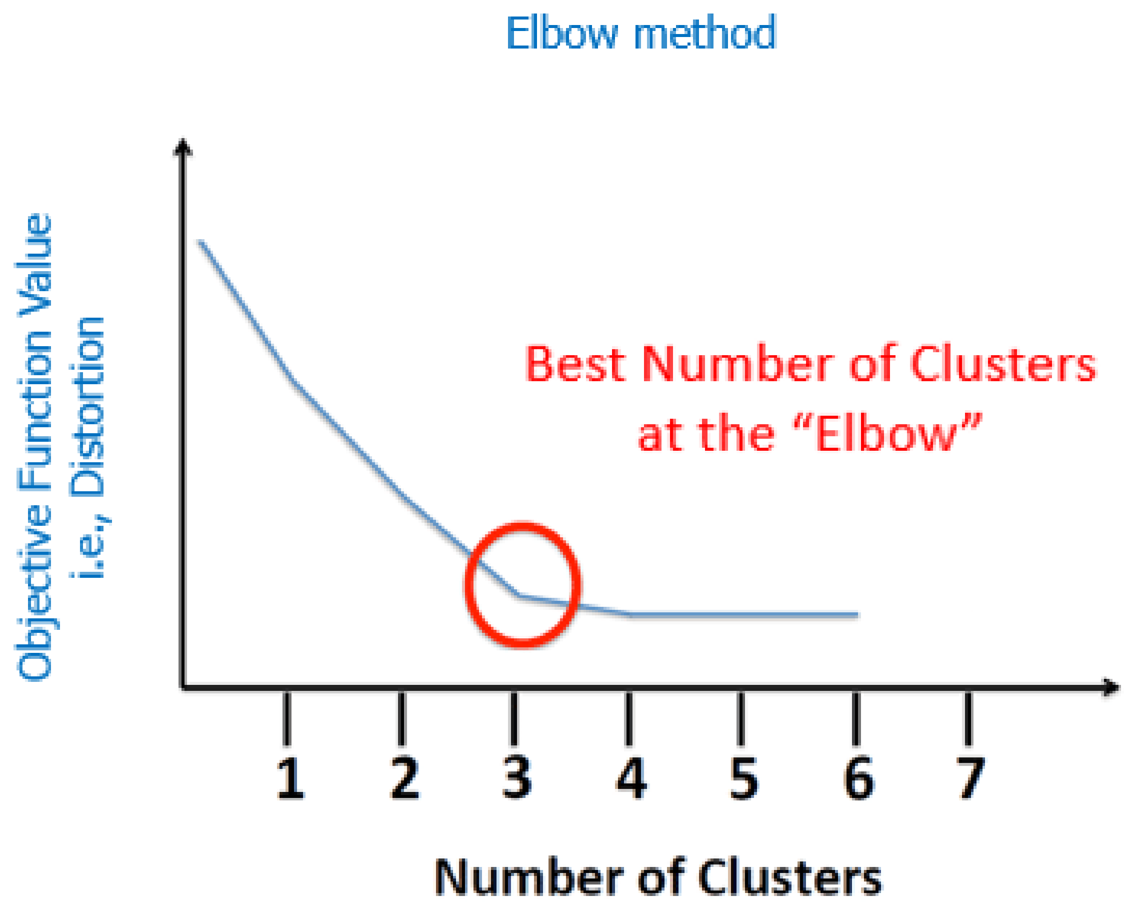
The Value of K

- One way to select K for the K-means algorithm is to try different values of K, plot the K-means objective versus K, and look at the "elbow-point" in the plot



- For the above plot, $K = 2$ is the elbow point

The Value of K



Limitations in K-means Clustering

Limitations of K-means

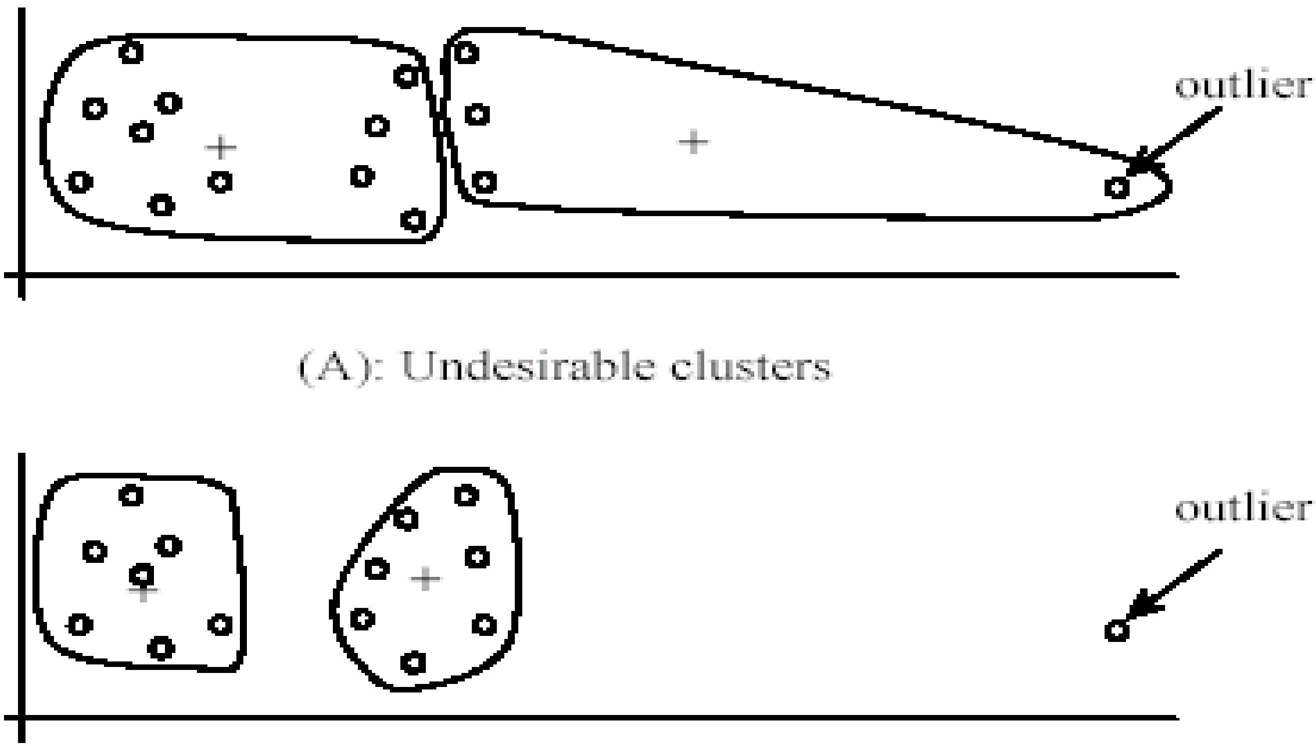
- K-means has problems when the data contains outliers

- *The K-means algorithm is very sensitive to the* ***initial seeds****.*

- K-means has problems when clusters are of different
  - Sizes
  - Densities
  - Non-globular shapes

Limitations of K-means

- K-means has problems when the data contains outliers

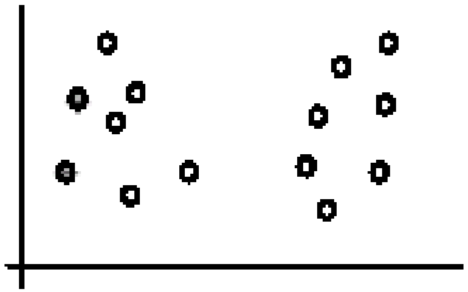


(A): Undesirable clusters

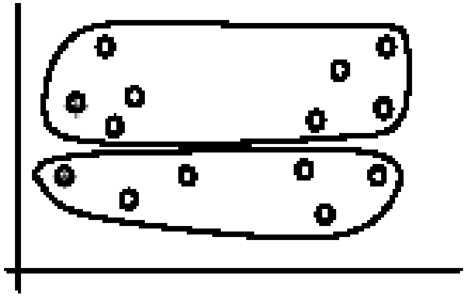
(B): Ideal clusters

Limitations of K-means

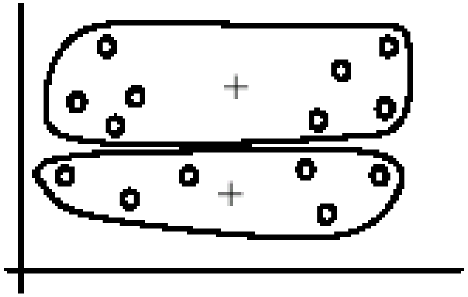
- The algorithm is sensitive to initial seeds



(A). Random selection of seeds (centroids)



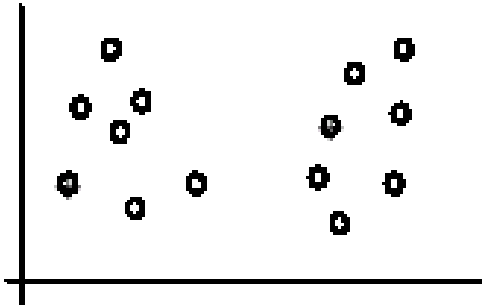
(B). Iteration 1



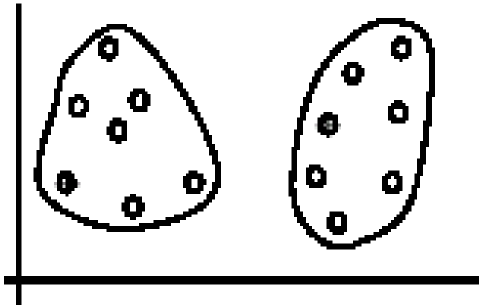
(C). Iteration 2

Limitations of K-means

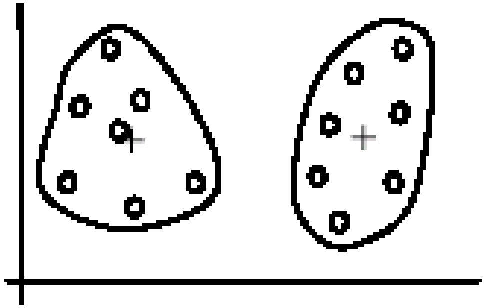
- The algorithm is sensitive to initial seeds



(A). Random selection of k seeds (centroids)



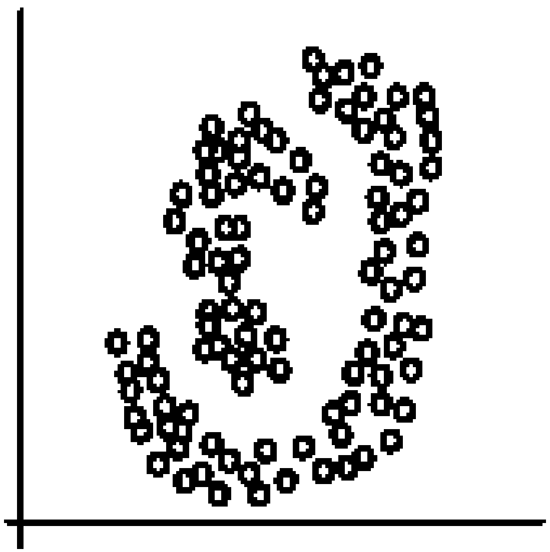
(B). Iteration 1



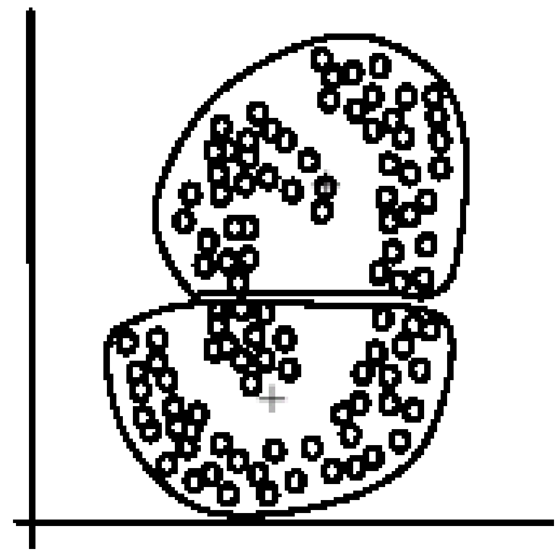
(C). Iteration 2

Limitations of K-means

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B): *k*-means clusters

K-Medoids Clustering

K-Medoids Clustering

- The k-means algorithm is sensitive to outliers!
  - Since an object with an extremely large value may substantially distort the distribution of the data.

K-Medoids:

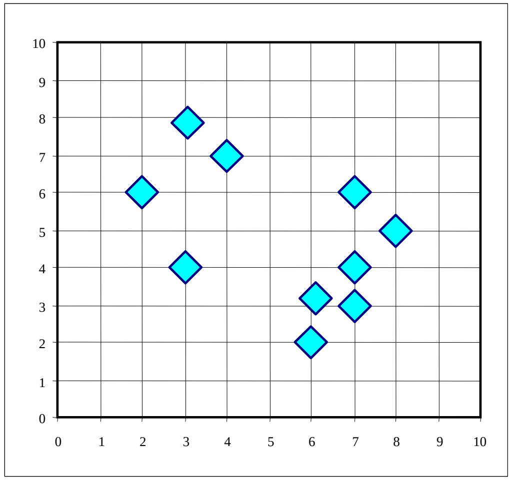
- Instead of taking the mean value of the object in a cluster as a reference point, *medoids can be used*, which is the most centrally located object in a cluster.

K-Medoids Clustering

- Find *representative* objects, called **medoids**, in the clusters

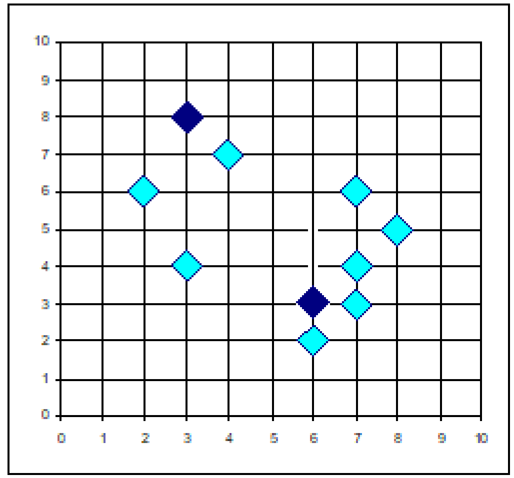
- ***PAM* (Partitioning Around Medoids, 1987)**
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets

K-Medoids Clustering



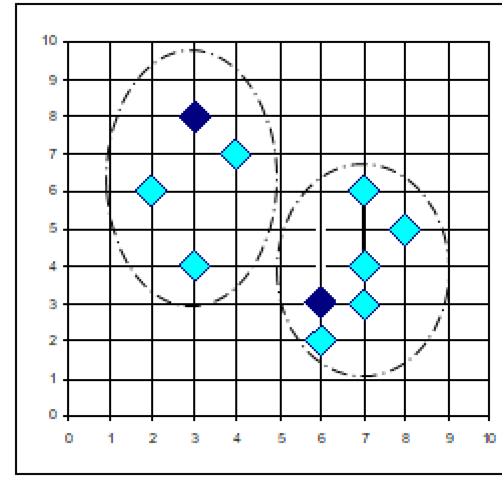
K=2

Arbitrary choose k object as initial medoids

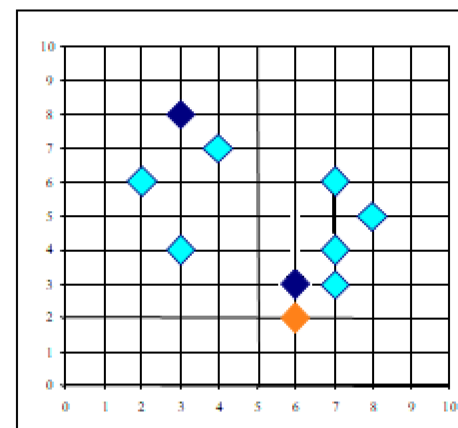


Assign each remaining object to nearest medoids

Total Cost = 20

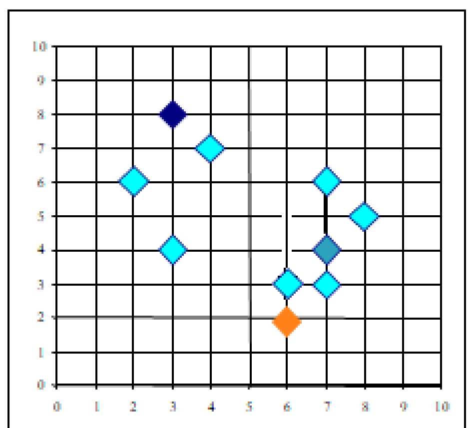


Randomly select a nonmedoid object, O_{ramdom}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

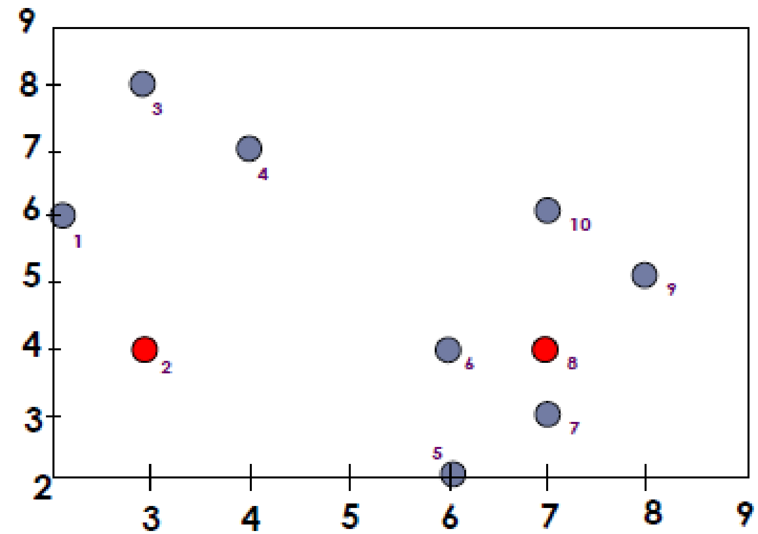
K-Medoids Clustering

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

K-Medoids Clustering

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



Goal: create two clusters

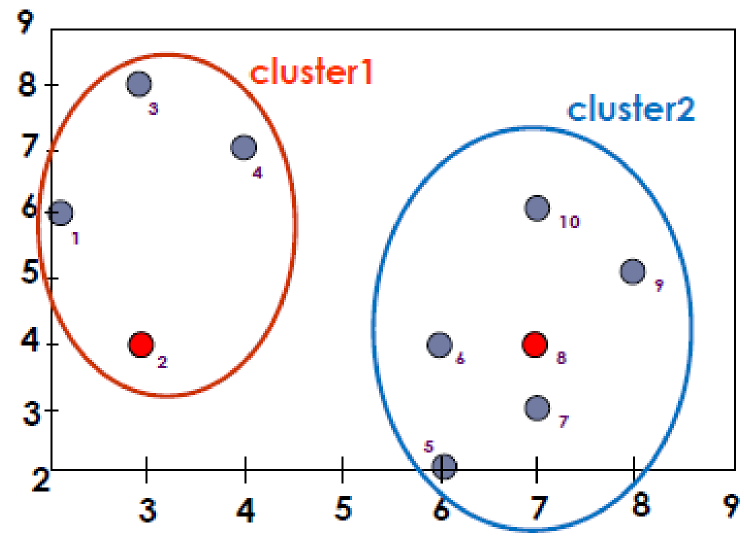
Choose randmly two medoids

$O_2 = (3,4)$
$O_8 = (7,4)$

K-Medoids Clustering

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Assign each object to the closest representative object

→Using L1 Metric (Manhattan), we form the following clusters

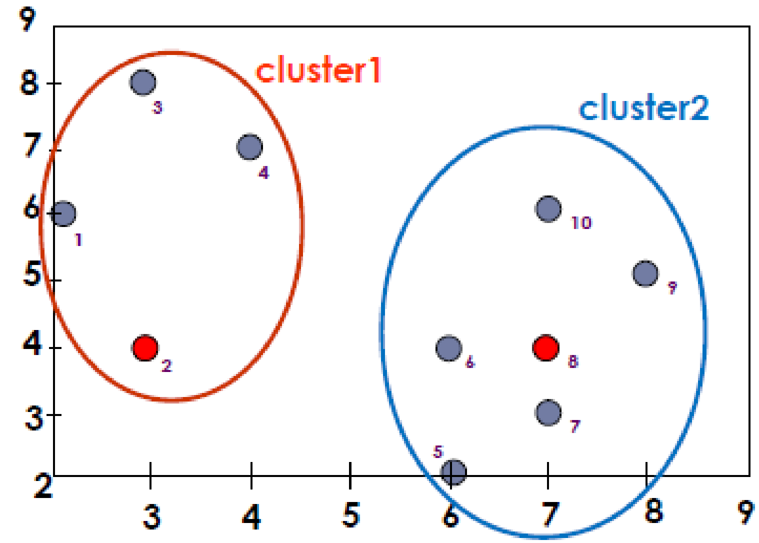
Cluster1 = $\{O_1, O_2, O_3, O_4\}$

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

K-Medoids Clustering

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Compute the absolute error criterion **[for the set of Medoids (O2,O8)]**

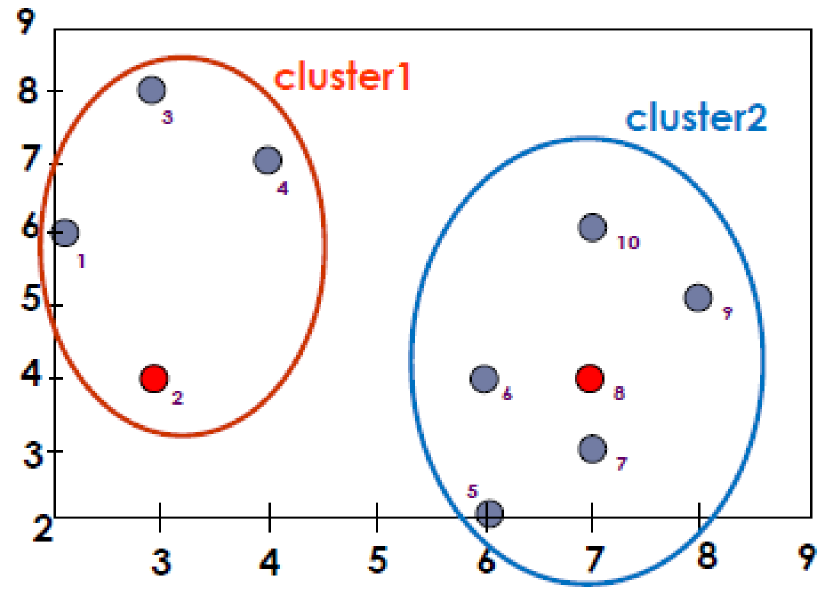
$$E=\sum_{i=1}^{k}\sum_{p\in C_i}|p-o_i| = |o_1-o_2|+|o_3-o_2|+|o_4-o_2|$$

$$+|o_5-o_8|+|o_6-o_8|+|o_7-o_8|+|o_9-o_8|+|o_{10}-o_8|$$

K-Medoids Clustering

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



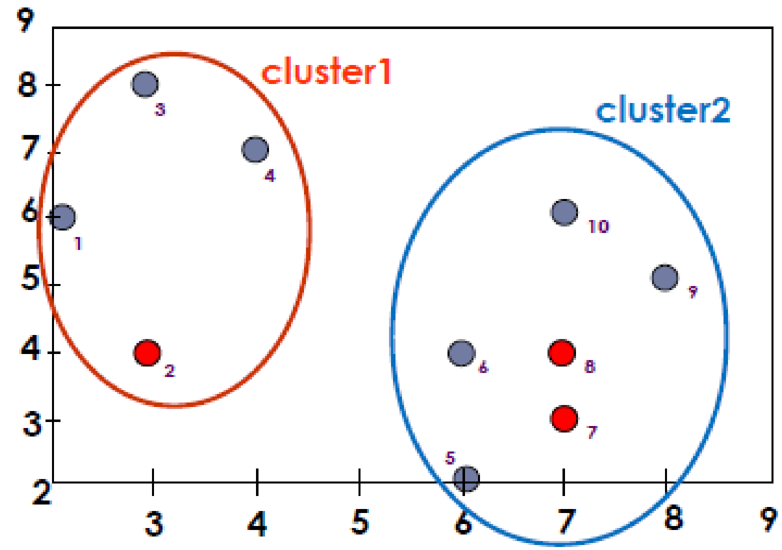
→The absolute error criterion **[for the set of Medoids (O2,O8)]**

$$E = (3+4+4)+(3+1+1+2+2) = 20$$

K-Medoids Clustering

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Choose a random object O_7

→Swap O8 and O7

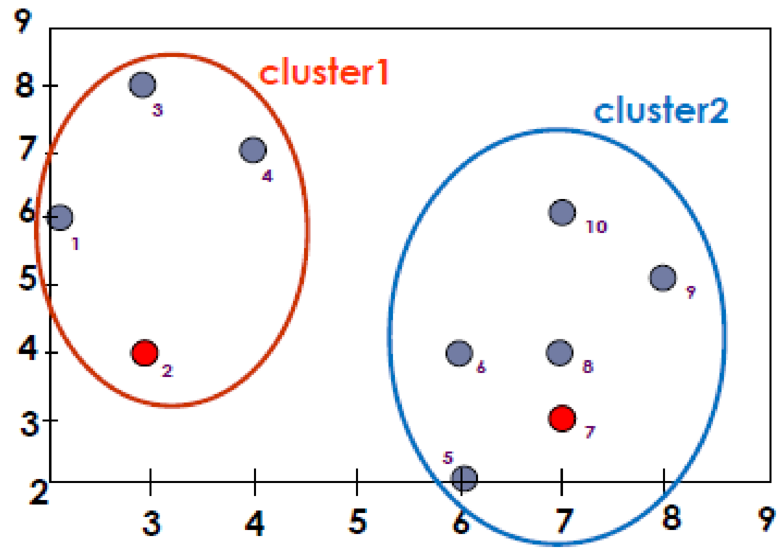
→Compute the absolute error criterion [for the set of Medoids (O2,O7)]

$$E = (3+4+4)+(2+2+1+3+3) = 22$$

K-Medoids Clustering

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Compute the cost function

Absolute error [for O_2,O_7] – Absolute error [O_2,O_8]

$$S = 22 - 20$$

$S > 0 \Rightarrow$ it is a bad idea to replace O_8 by O_7

K-Medoids Clustering

- *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 points,
  - k is # of clusters