Clustering (K-Medoids)

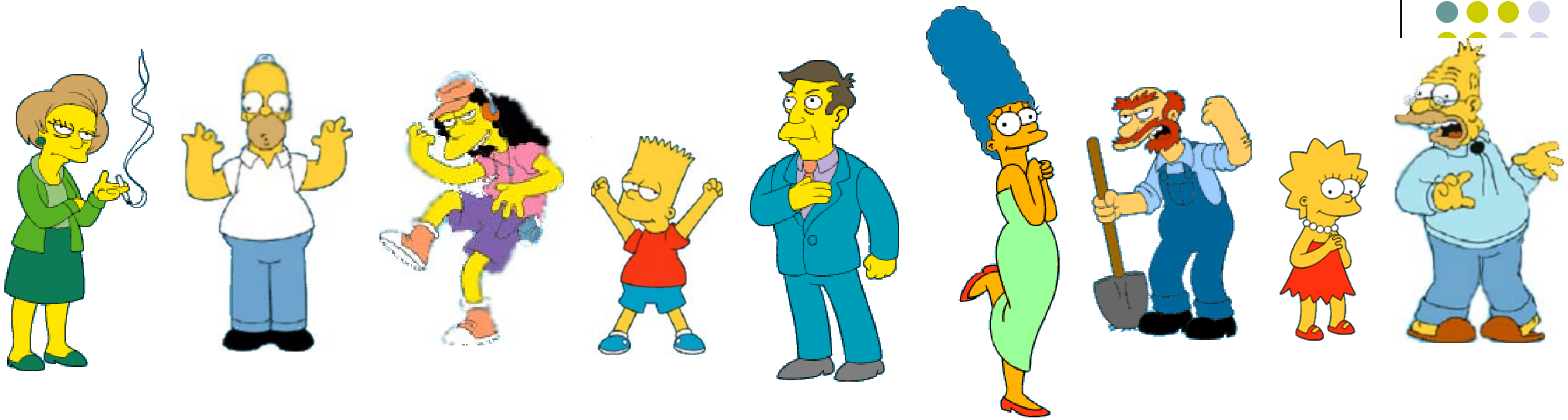
Dr. Md. Golam Rabiul Alam

What is Clustering?

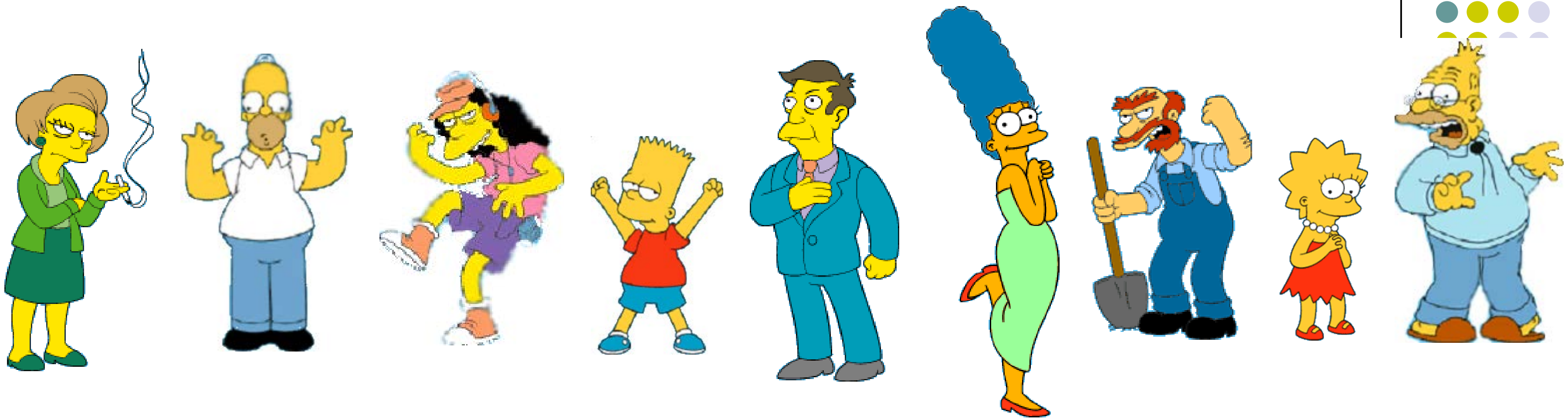
Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in that group and dissimilar to the data points in other groups.

- Organizing data into groups such that there is
  - high intra-group similarity
  - low inter-group similarity
- Finding the class labels and the number of classes directly from the data (in contrast to classification).
- More informally, finding natural groupings among objects.

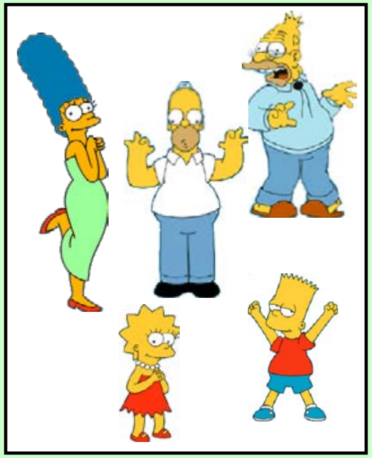
What is a natural grouping among these objects?



What is a natural grouping among these objects?

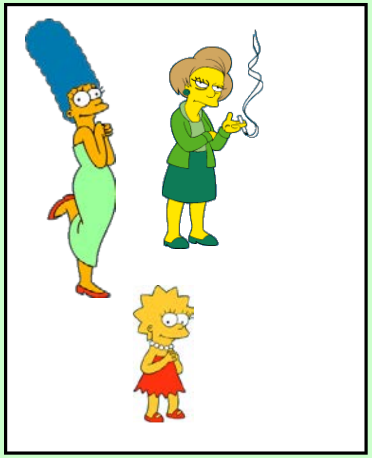


Clustering is subjective



Simpson's Family

School Employees



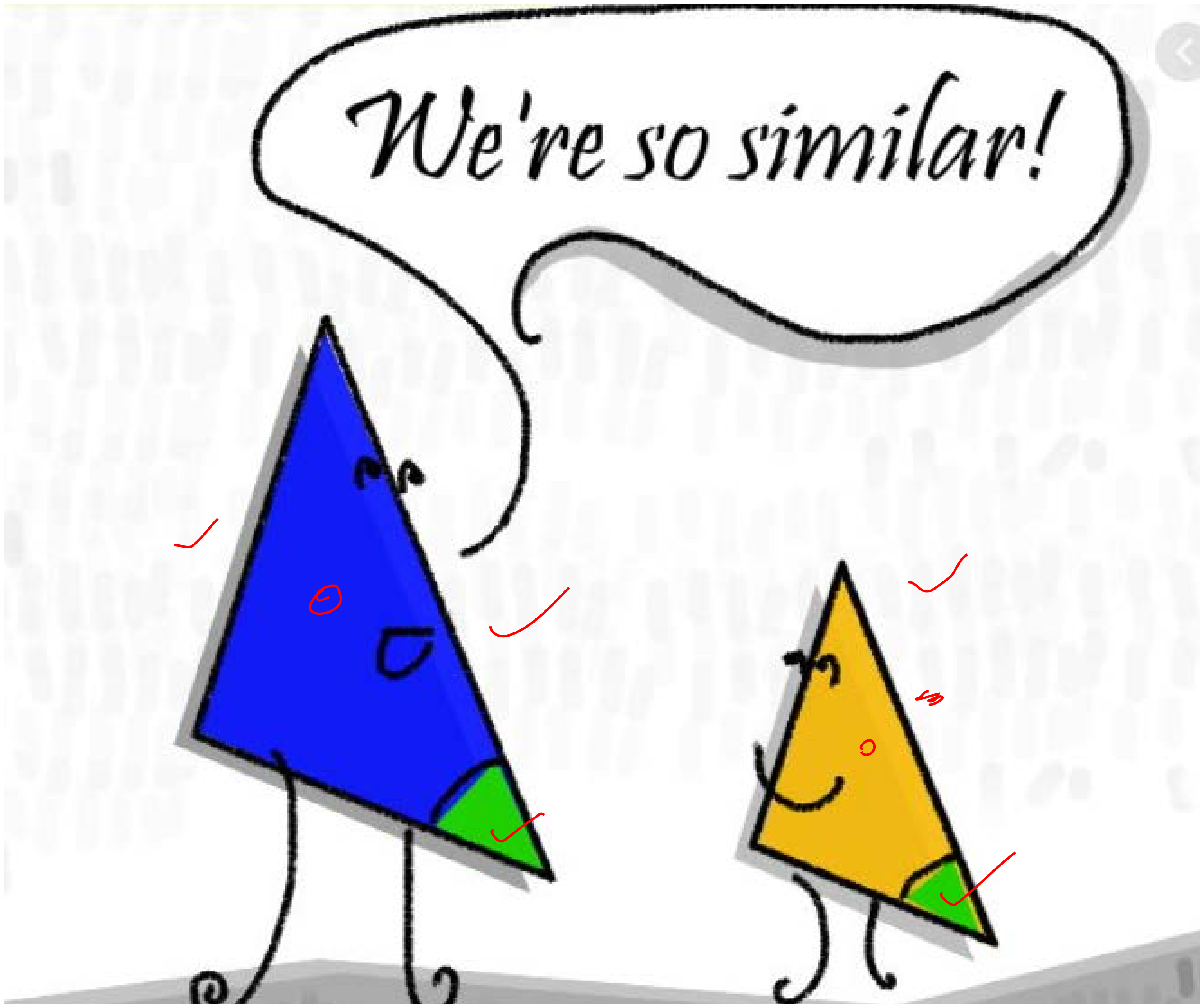
Females

Males

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features.

Webster's Dictionary

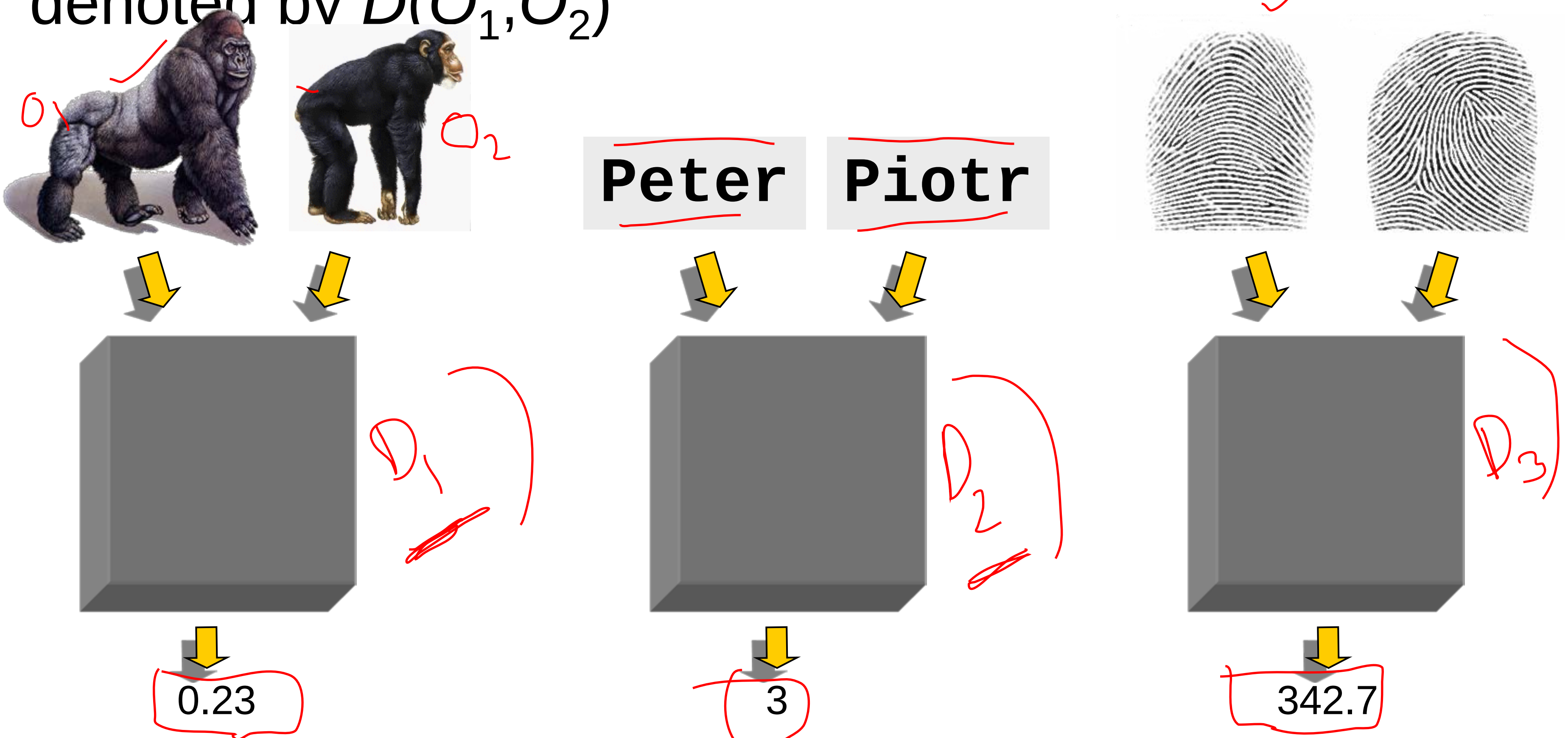


Similarity is hard to define, but…
"*We know it when we see it*"

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

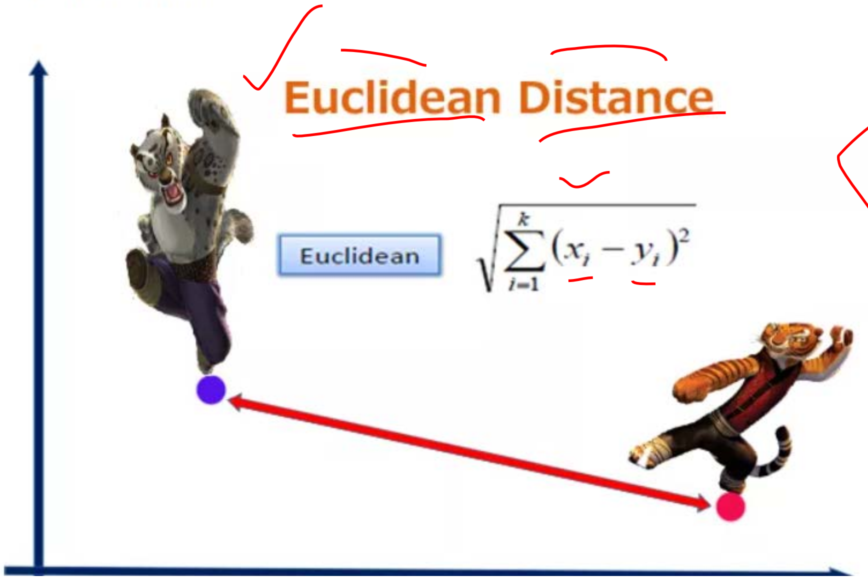
Defining Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$



Common Distance Functions

Euclidean distance:



$\sqrt{(0-7)^2+(3-6)^2+(4-3)^2+(5-(-1))^2}$

Euclidean distance implementation in python:

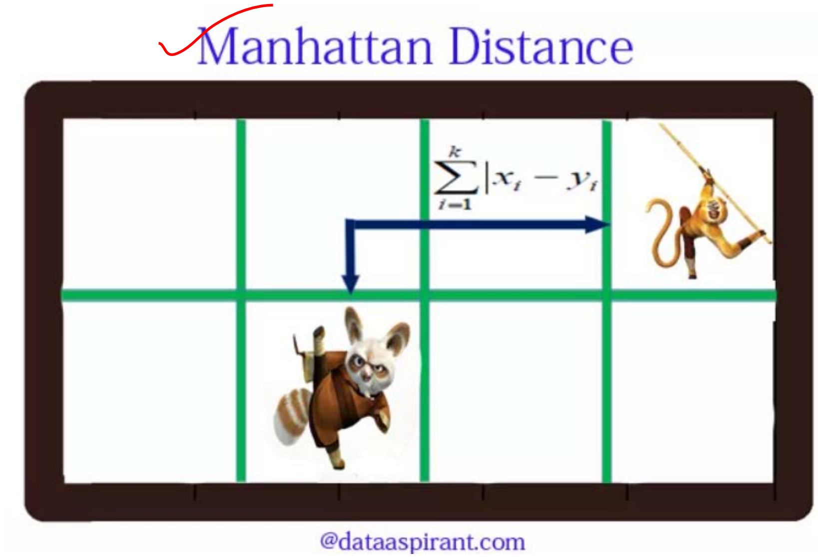
```
#!/usr/bin/env python


from math import*

def euclidean_distance(x,y):

    return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))

print euclidean_distance([0,3,4,5],[7,6,3,-1])
```

Script Output:

```
9.74679434481
[Finished in 0.0s]
```

Common Distance Functions

Manhattan distance:



In a plane with p1 at (x1, y1) and p2 at (x2, y2).

Manhattan distance = |x1 – x2| + |y1 – y2|

|10 - 10| + |20 - 20| + |10 - 20|

Manhattan distance implementation in python:

```
#!/usr/bin/env python

from math import*

def manhattan_distance(x,y):

    return sum(abs(a-b) for a,b in zip(x,y))

print manhattan_distance([10,20,10],[10,20,20])
```

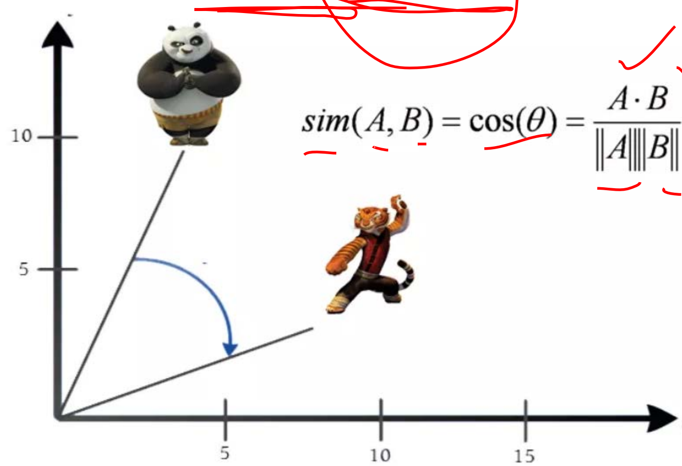
Script Output:

```
10
[Finished in 0.0s]
```

Common Distance Functions

Cosine similarity:

Cosine Similarity



$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

$3*2 + 45*54 + 7*13 + 2*15$

$\sqrt{3^2 + 45^2 + 7^2 + 2^2} * \left(\sqrt{2^2 + 54^2 + 13^2 + 15^2}\right)$

Cosine similarity implementation in python:

```
#!/usr/bin/env python


from math import*

def square_rooted(x):

    return round(sqrt(sum([a*a for a in x])),3)

def cosine_similarity(x,y):

    numerator = sum(a*b for a,b in zip(x,y))
   denominator = square_rooted(x)*square_rooted(y)
   return round(numerator/float(denominator),3)

print cosine_similarity([3, 45, 7, 2], [2, 54, 13, 15])
```

Script Output:

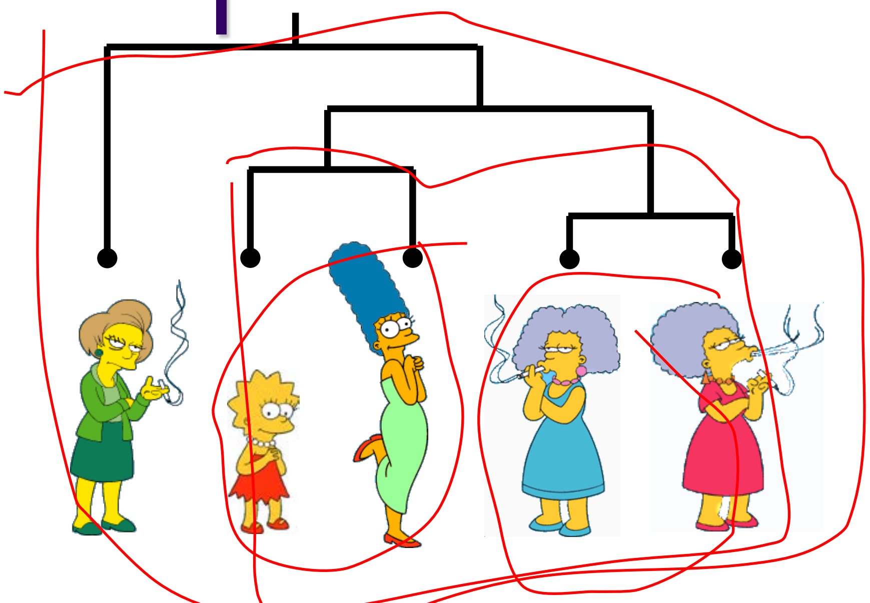
```
0.972
[Finished in 0.1s]
```

Two Types of Clustering

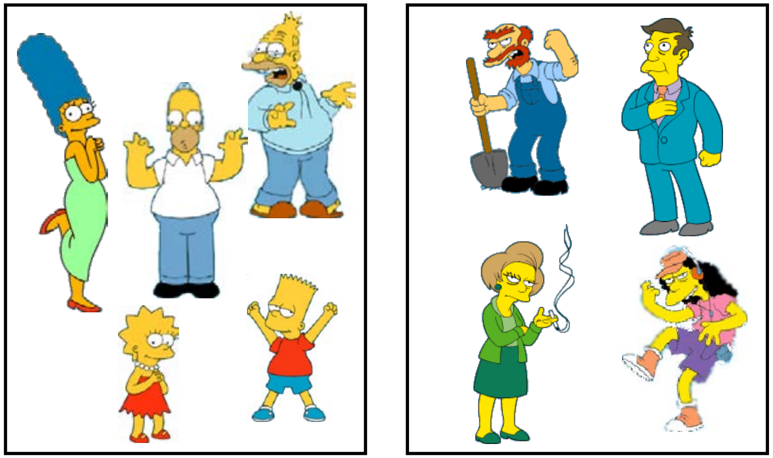
• **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion (k-means, k-medoids).

•**Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion (BIRCH, CAMELEON).

Hierarchica
l



Partitional



K-MEANS CLUSTERING

K-Means

- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2,$$

where x_n is a vector representing the n^{th} data point and μ_j is the geometric centroid of the data points in S_j(set of data points in cluster j).

Algorithm *k-means*

1. Decide on a value for *k*.

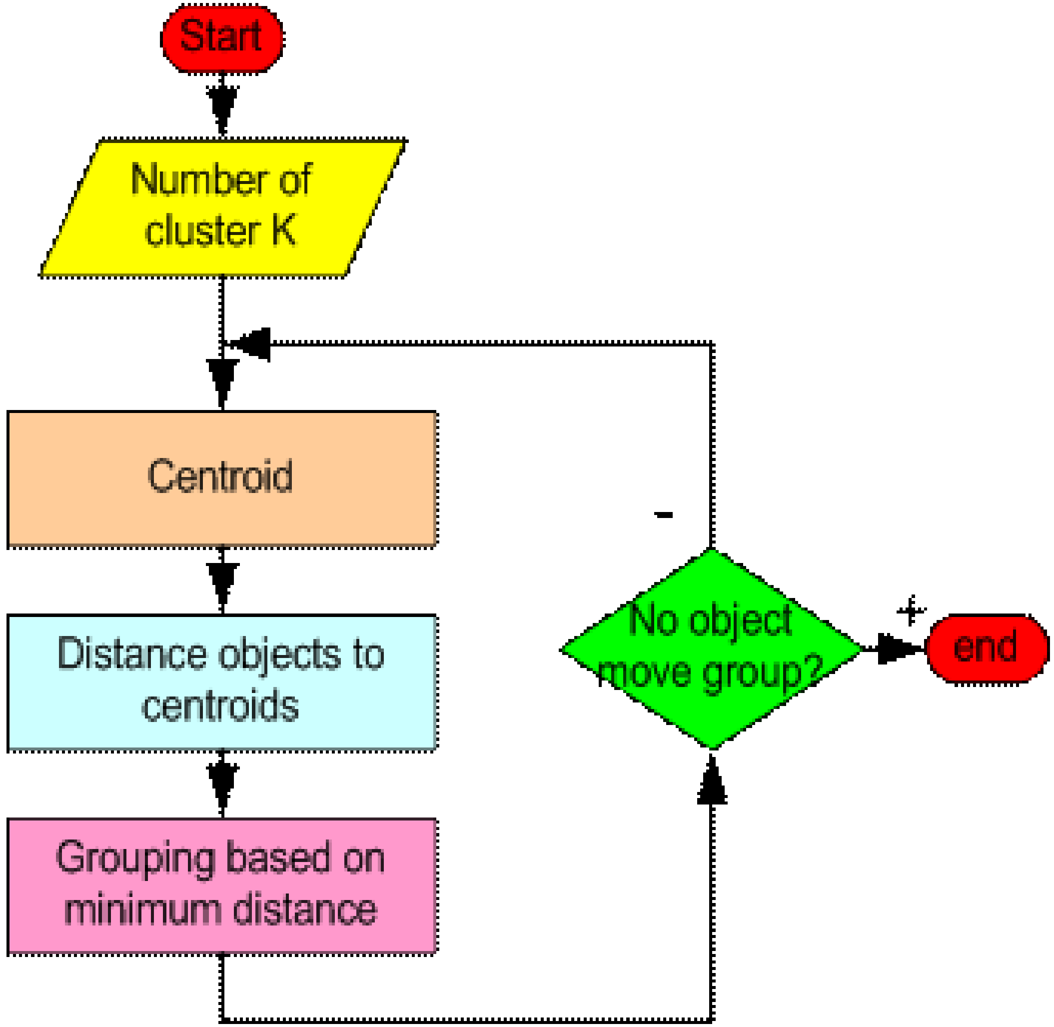
2. Initialize the *k* cluster centers (randomly, if necessary).

3. Decide the memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.

5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise goto 3.

How the K-Mean Clustering algorithm works?



A Simple example of k-means clustering (using K=2)

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

<u>Step 1</u>:

<u>Initialization</u>: Randomly we choose following two centroids (k=2) for two clusters.

In this case the 2 centroids are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 2:

$m_1(1.0, 1.0)$ $m_2(5.0, 7.0)$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

m1=(1.0,1.0) and m2=(5.0,7.0)

$$d(m_1,2) = \sqrt{|1.0-1.5|^2 + |1.0-2.0|^2} = 1.12$$

$$d(m_2,2) = \sqrt{|5.0-1.5|^2 + |7.0-2.0|^2} = 6.10$$

$$\sqrt{(1-5)^2 + (1-7)^2}$$

$$= 7.21$$

Step 2:

- Thus, we obtain two clusters containing:

  {1,2,3} and {4,5,6,7}.

- Their new centroids are:

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right)$$

$$= (4.12, 5.38)$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.

Centroid1: m1=(1.83, 2.33)

Centroid2: m2=(4.12, 5.38)

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

- Therefore, the new clusters are:

  {1,2} and {**3**,4,5,6,7}

$m_1 \left(\frac{1+1.5}{2}, \frac{1+2}{2} \right)$

- Next centroids are: m1=(1.25,1.5) and m2 = (3.9,5.1)

$m_2 \left(\frac{3+5+3.5+4.5+3.5}{5}, \frac{4+7+5+5+4.5}{5} \right)$

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

- <u>Step 4</u> :

  Considering the obtained centroids i.e., m1=(1.25,1.5) and m2 = (3.9,5.1)

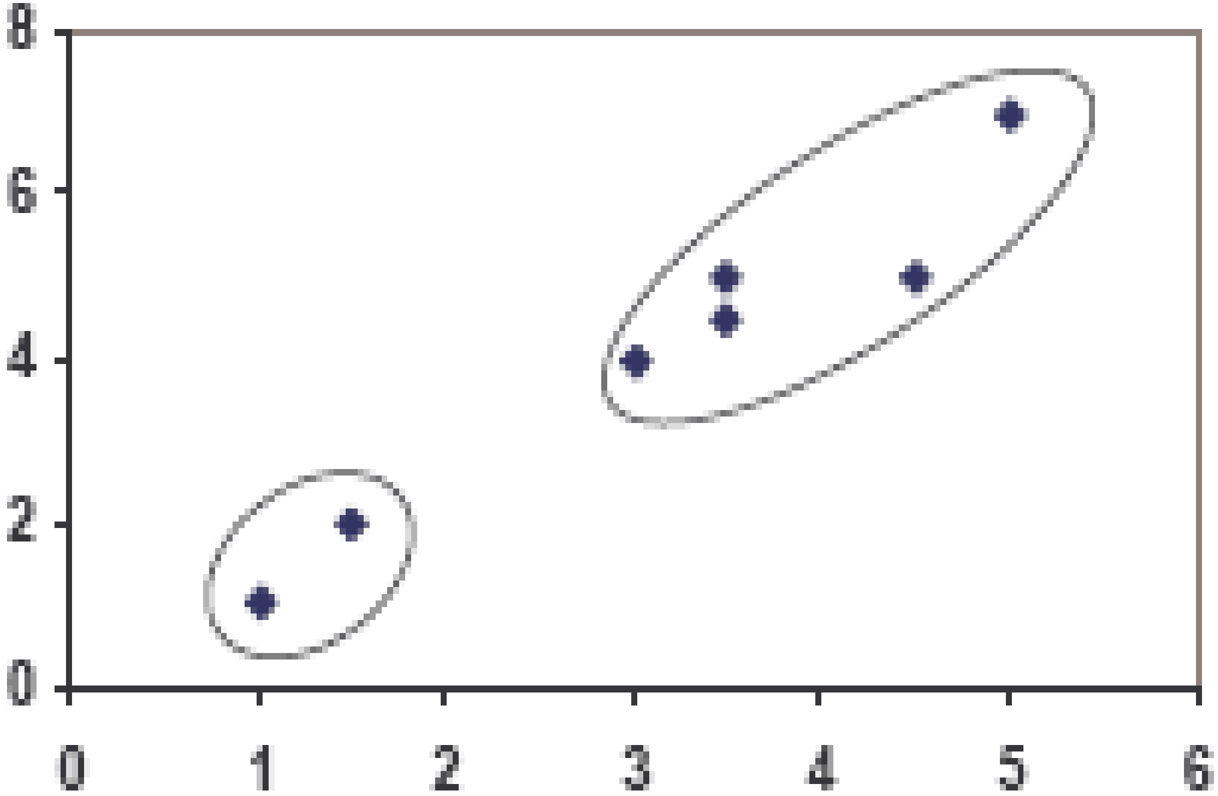
The obtained clusters are:

{1,2} and {3,4,5,6,7}

- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0.56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

PLOT



(with K=3)

Individual	$m_1 = 1$	$m_2 = 2$	$m_3 = 3$	cluster
1	0	1.11	3.61	1
2	1.12	0	2.5	2
3	3.61	2.5	0	3
4	7.21	6.10	3.61	3
5	4.72	3.61	1.12	3
6	5.31	4.24	1.80	3
7	4.30	3.20	0.71	3

C_3

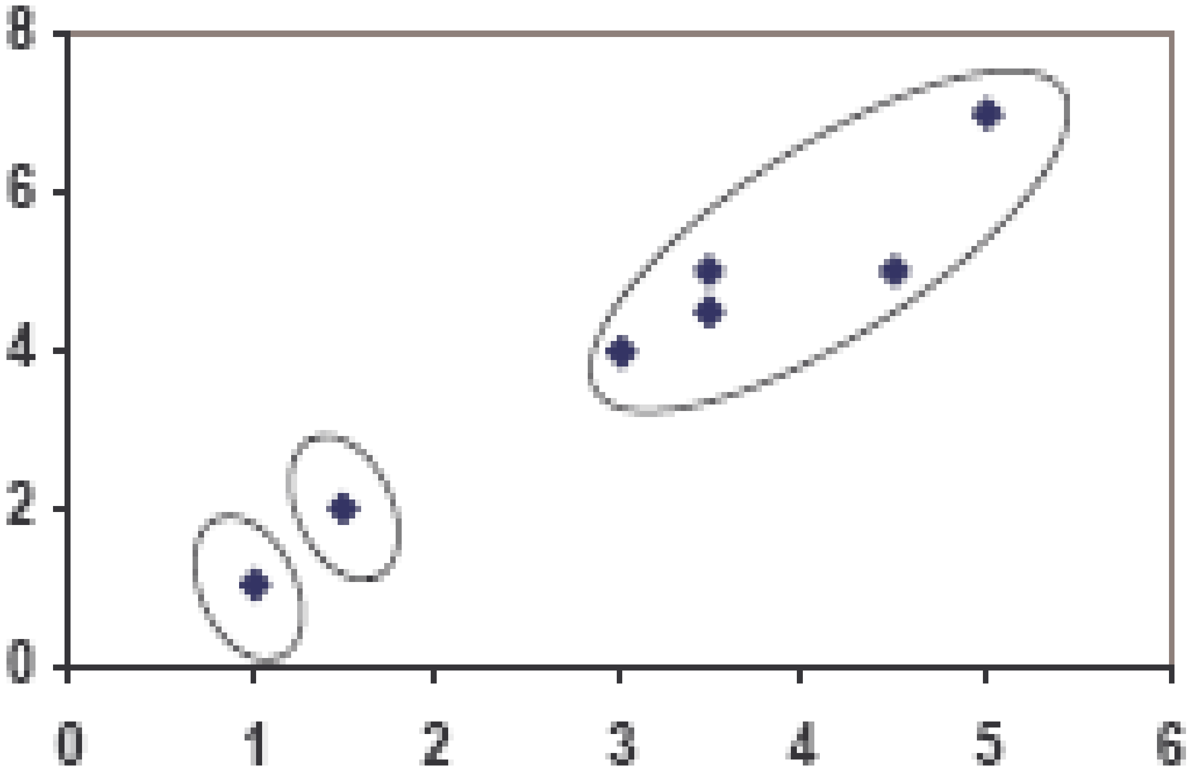
clustering with initial centroids (1, 2, 3)

Step 1

Individual	m_1 (1.0, 1.0)	m_2 (1.5, 2.0)	m_3 (3.9,5.1)	cluster
1	0	1.11	5.02	1
2	1.12	0	3.92	2
3	3.61	2.5	1.42	3
4	7.21	6.10	2.20	3
5	4.72	3.61	0.41	3
6	5.31	4.24	0.61	3
7	4.30	3.20	0.72	3

Step 2

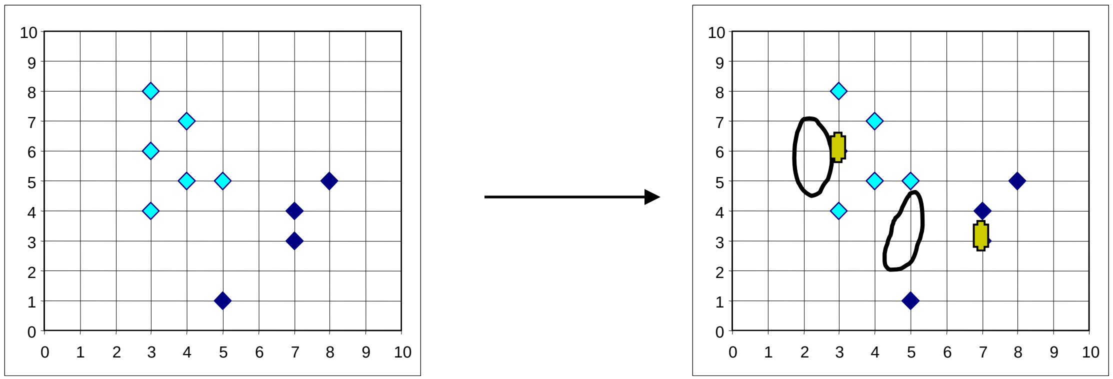
PLOT



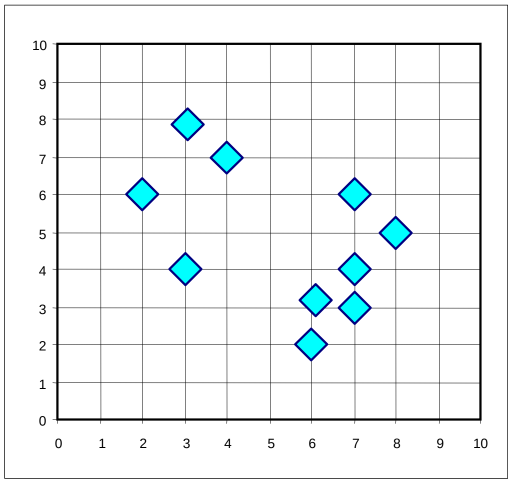
K-MEDOIDS CLUSTERING

What Is the Problem of the K-Means Method?

- 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** (*representative* objects) object in a cluster

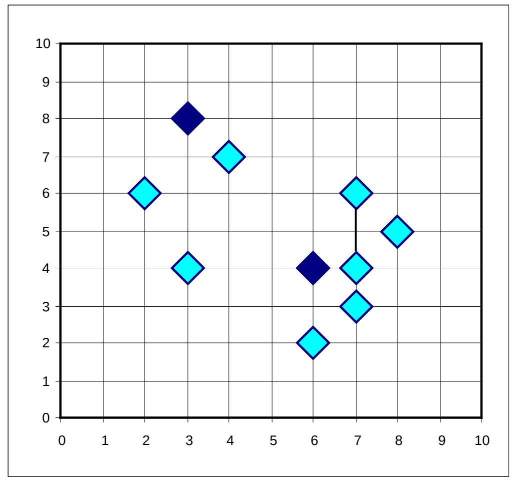


PAM: A Typical K-Medoids Algorithm



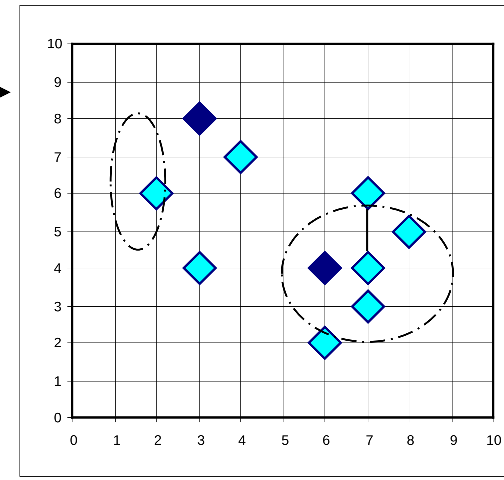
K=2

Arbitrary choose k object as initial medoids

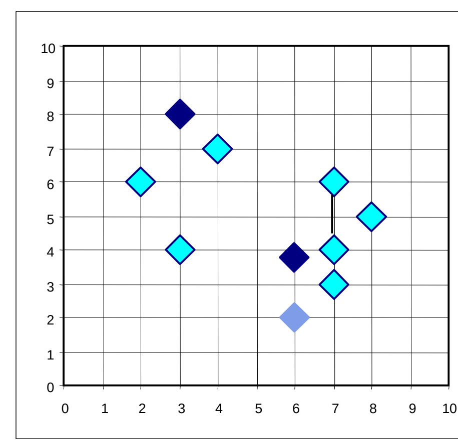


Assign each remainin g object to nearest medoids

Total Cost = 20

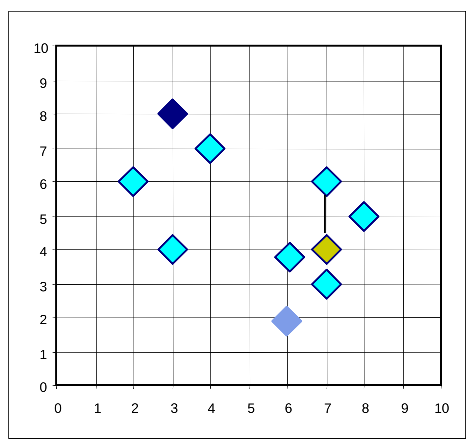


Randomly select a non-medoid object, O_{random}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

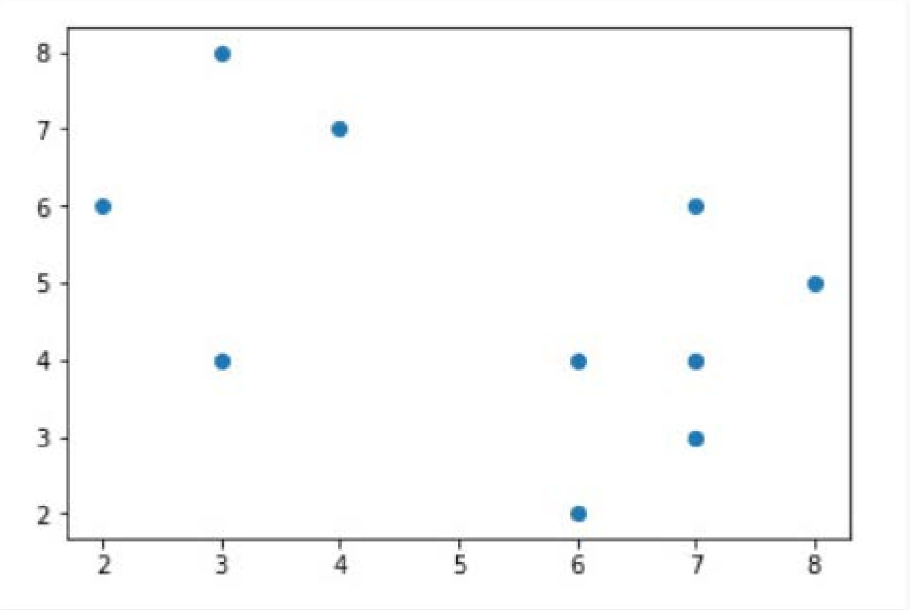
If quality is improved.

Do loop

Until no change

K-Medoids Clustering Example

	X	Y
0	7	6
1	2	6
2	3	8
3	8	5
4	7	4
5	4	7
6	6	2
7	7	3
8	6	4
9	3	4



K-Medoids Clustering Example

Step #1: k = 2

Let the randomly selected 2 medoids be `c1 -(3, 4)` and `c2 -(7, 4)`.

Step #2: Calculating cost.

The dissimilarity of each non-medoid point with the medoids is calculated and tabulated:

	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	7	6	6	2
1	2	6	3	7
2	3	8	4	8
3	8	5	6	2
4	7	4	4	0
5	4	7	4	6
6	6	2	5	3
7	7	3	5	1
8	6	4	3	1
9	3	4	0	4

Each point is assigned to the cluster of that medoid whose dissimilarity is less.
The points 1, 2, 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2.
The cost C = (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2)
C = 20

K-Medoids Clustering Example

Step #3: Now randomly select one non-medoid point and recalculate the cost.

Let the randomly selected point be (7, 3). The dissimilarity of each non-medoid point with the medoids – `c1 (3, 4)` and `c2 (7, 3)` is calculated and tabulated.

	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	7	6	6	3
1	2	6	3	8
2	3	8	4	9
3	8	5	6	3
4	7	4	4	1
5	4	7	4	7
6	6	2	5	2
7	7	3	-	-
8	6	4	3	2
9	3	4	-	-

Each point is assigned to that cluster whose dissimilarity is less. So, the points 1, 2, 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2.

The cost C = (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3)

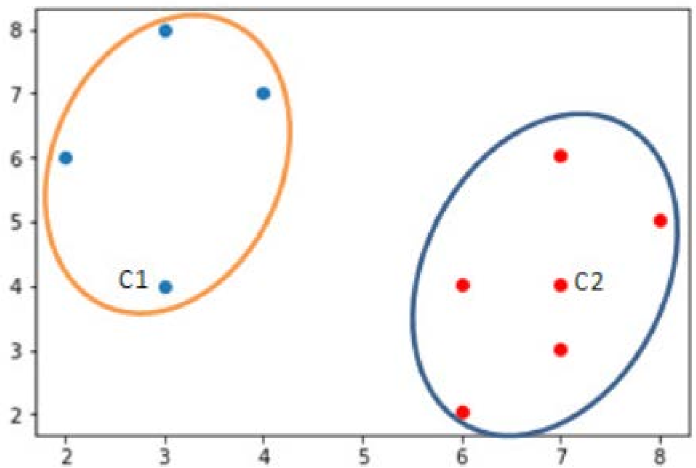
C = 22

Swap Cost = Present Cost – Previous Cost

= 22 – 20 = 2 >0

As the swap cost is not less than zero, we undo the swap. Hence (3, 4) and (7, 4) are the final medoids.

K-Medoids Clustering Example



The dissimilarity of the `medoid(Ci)` and `object(Pi)` is calculated by using `E = |Pi - Ci|`

The cost in K-Medoids algorithm is given as –

$$c = \sum_{Ci} \sum_{Pi \in Ci} |Pi - Ci|$$

Algorithm:

1. Initialize: select k random points out of the n data points as the medoids.

2. Associate each data point to the closest medoid by using any common distance metric methods.

3. While the cost decreases:

For each medoid m, for each data o point which is not a medoid:

1. Swap m and o, associate each data point to the closest medoid, recompute the cost.

2. If the total cost is more than that in the previous step, undo the swap.

The **time complexity** is $O(k * (n - k)^2)$

The K-Medoid Clustering Method

Advantages:

1. It is simple to understand and easy to implement.
2. K-Medoid Algorithm is fast and converges in a fixed number of steps.
3. PAM is less sensitive to outliers than other partitioning algorithms.

Disadvantages:

1. The main disadvantage of K-Medoid algorithms is that it is not suitable for clustering non-spherical (arbitrary shaped) groups of objects. This is because it relies on minimizing the distances between the non-medoid objects and the medoid (the cluster center) – briefly, it uses compactness as clustering criteria instead of connectivity.
2. It may obtain different results for different runs on the same dataset because the first k medoids are chosen randomly.

The K-Medoid Clustering Method

- *K-Medoids* Clustering: Find *representative* objects (<u>medoids</u>) in clusters
  - *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 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 (due to the computational complexity)
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling

CONCLUSION

- *K-means and K-medoids algorithms are* useful for undirected knowledge discovery and relatively simple.
- Both of the algorithms have wide spread usage in lot of fields, ranging from pattern recognitions, image processing, machine vision, and many others.

References

- Tutorial - Tutorial with introduction of Clustering Algorithms (k-means, fuzzy-c-means, hierarchical, mixture of gaussians) + some interactive demos (java applets).
- Digital Image Processing and Analysis-byB.Chanda and D.Dutta Majumdar.
- H. Zha, C. Ding, M. Gu, X. He and H.D. Simon. "Spectral Relaxation for K-means Clustering", Neural Information Processing Systems vol.14 (NIPS 2001). pp. 1057-1064, Vancouver, Canada. Dec. 2001.
- J. A. Hartigan (1975) "Clustering Algorithms". Wiley.
- J. A. Hartigan and M. A. Wong (1979) "A K-Means Clustering Algorithm", Applied Statistics, Vol. 28, No. 1, p100-108.
- D. Arthur, S. Vassilvitskii (2006): "How Slow is the k-means Method?,"
- D. Arthur, S. Vassilvitskii: "k-means++ The Advantages of Careful Seeding" 2007 Symposium on Discrete Algorithms (SODA).
- https://www.geeksforgeeks.org/ml-k-medoids-clustering-with-example/

Thank You