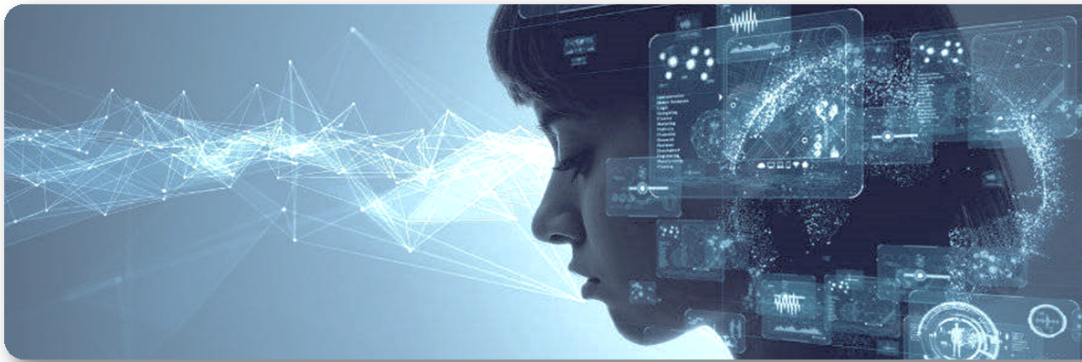


COMP 2211 Exploring Artificial Intelligence

K-Means Clustering

Dr. Desmond Tsoi

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology, Hong Kong SAR, China

Problem

- Given the following simplified Mall customers dataset with attributes, age, income (in thousands) and expense score (1-100).

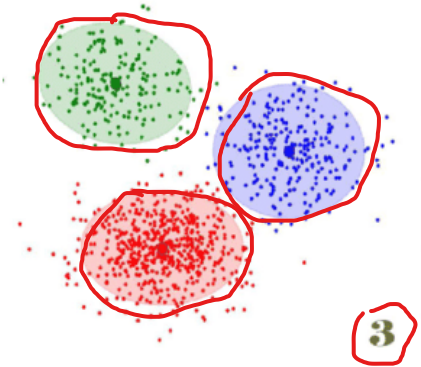
Person	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Age	19	67	35	60	65	49	70	70	57	68	23	65	27	47	57	43	56	40	37	34
Income $\times$1000	15	19	24	30	38	42	46	49	54	59	62	63	67	71	75	78	79	87	97	103
Expense score 1-100	39	14	35	4	35	52	56	55	51	55	41	52	56	9	5	17	35	13	32	23

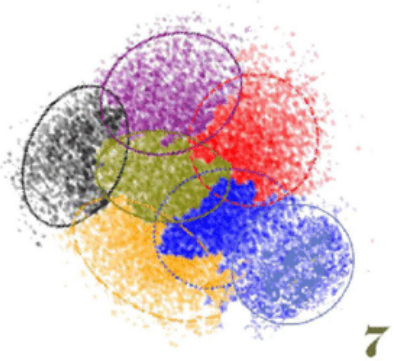
Note: Expense score goes from 1 (low spends) to 100 (high spends).

- Could we cluster the customers based on their characteristics (age, income & expense score) to determine the targeted advertisement and make the marketing budget more efficient?

What is Clustering?

- Clustering is grouping things into "natural" categories when no class label is available.

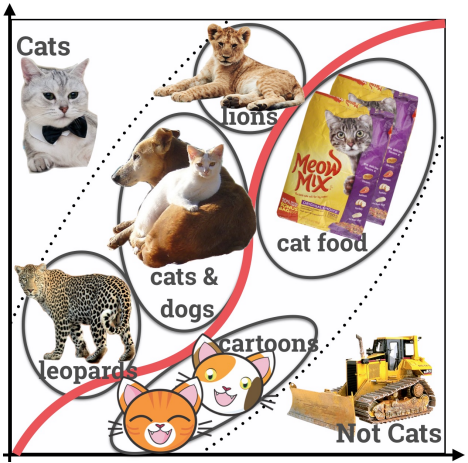




Why Clustering?

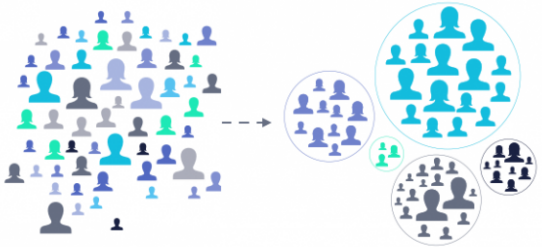
- Labeling a large set of data samples can be costly.
- Clustering can be used for finding features that will be useful later for categorization.

auto-labeling.



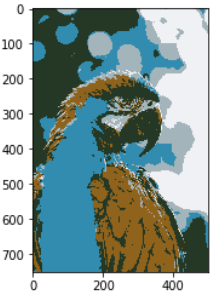
What is Clustering for?

- Group people of similar sizes together to make "small", "medium", and "large" T-shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: Does not fit all
- Given a collection of text documents, we want to organize them according to their content similarities
  - To produce a topic hierarchy
- In marketing, segment customers according to their similarities
  - To do targeted marketing

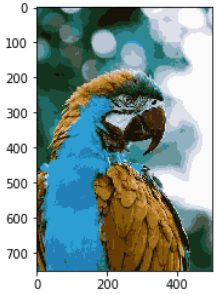


What is Clustering for?

- Image segmentation
  - To partition a digital image into multiple image segments, also known as image regions.



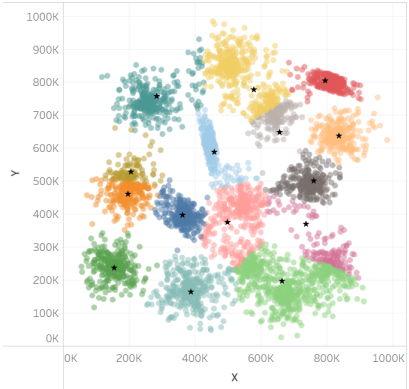
k = 5



k = 20

K-Means Clustering

- K-Means clustering is an unsupervised learning algorithm.
- There is no labeled data for this clustering, unlike supervised learning.
- It performs the division of data into non-overlapping clusters that share similarities and are dissimilar to the data belonging to another cluster.
- The term 'K' is a number telling the system how many clusters you need to create. For example, if K = 3 refers to three clusters.

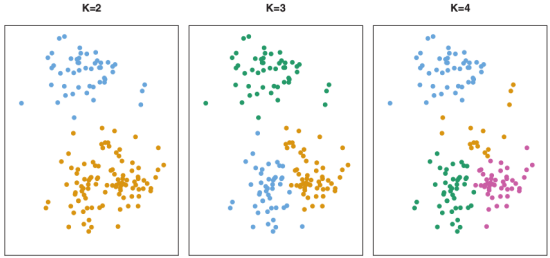


K-Means Clustering

- Let the set of data points

$$\mathbf{D} = \{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_n}\}$$

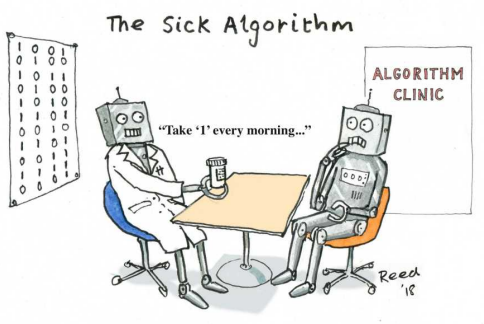
  where $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{id})$ is the i^{th} data point, and d is the number of dimensions.
- To perform K-means clustering, we must first specify the desired number of clusters K.
- The K-Means algorithm partitions the given data points into K non-overlapped clusters:
  - Each cluster has a cluster center, called centroid



K-means Algorithm

Given K, the K-Means algorithm works as follows:

1. Choose K (random) data points (seeds) to be the initial centroids (cluster centers)
2. Find the distances between each data point in our training set with the K centroids
3. Assign each data point to the closest centroid according to the distance found
4. Re-compute the centroids using the current cluster memberships
5. If a convergence criterion is NOT met, repeat steps 2 to 4



Example

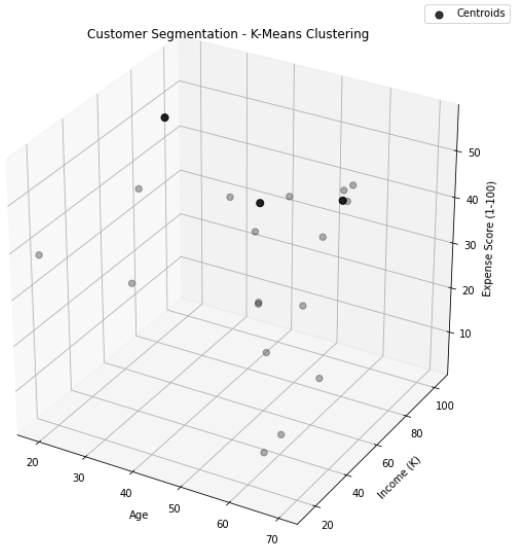
- Let's perform K-Means Clustering on the following simplified Mall customers dataset with attributes, age, income (in thousands) and expense score (1-100).

Person	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Age	19	67	35	60	65	49	70	70	57	68	23	65	27	47	57	43	56	40	37	34
Income $\times 1000$	15	19	24	30	38	42	46	49	54	59	62	63	67	71	75	78	79	87	97	103
Expense score 1-100	39	14	35	4	35	52	56	55	51	55	41	52	56	9	5	17	35	13	32	23

Note: Expense score goes from 1 (low spends) to 100 (high spends).

Step 1: Randomly Pick 3 Data Points as Initial Centroids

- 1st centroid: (70, 46, 56)
- 2nd centroid: (27, 67, 56)
- 3rd centroid: (37, 97, 32)



Step 2: Find the Distances Between Each Data Point with the 3 Centroids

- 1st centroid: (70, 46, 56)
- 2nd centroid: (27, 67, 56)
- 3rd centroid: (37, 97, 32)

Assume Euclidean distance is used.

Person	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Age	19	67	35	60	65	49	70	70	57	68	23	65	27	47	57	43	56	40	37	34
Income $\times 1000$	15	19	24	30	38	42	46	49	54	59	62	63	67	71	75	78	79	87	97	103
Expense score 1-100	39	14	35	4	35	52	56	55	51	55	41	52	56	9	5	17	35	13	32	23
DC1	62	50	46	55	23	22	0	3	16	13	52	18	48	58	60	57	42	67	65	75
DC2	55	75	49	72	52	34	48	47	33	42	16	38	0	51	60	44	38	49	40	49
DC3	84	86	73	76	65	60	65	63	51	54	39	48	40	36	40	25	26	22	0	11

where DC1, DC2, DC3 are the distances between the data points and 1st centroid, 2nd centroid, and 3rd centroid, respectively.

Step 3: Assign Each Data Point to the Closest Centroid

- 1st centroid: (70, 46, 56)
- 2nd centroid: (27, 67, 56)
- 3rd centroid: (37, 97, 32)

Person	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Age	19	67	35	60	65	49	70	70	57	68	23	65	27	47	57	43	56	40	37	34
Income $\times$1000	15	19	24	30	38	42	46	49	54	59	62	63	67	71	75	78	79	87	97	103
Expense score 1-100	39	14	35	4	35	52	56	55	51	55	41	52	56	9	5	17	35	13	32	23
DC1	62	50	46	55	23	22	0	3	16	13	52	18	48	58	60	57	42	67	65	75
DC2	55	75	49	72	52	34	48	47	33	42	16	38	0	51	60	44	38	49	40	49
DC3	84	86	73	76	65	60	65	63	51	54	39	48	40	36	40	25	26	22	0	11
Cluster	2	1	1	1	1	1	1	1	1	1	2	1	2	3	3	3	3	3	3	3

where DC1, DC2, DC3 are the distances between the data points and 1st centroid, 2nd centroid, and 3rd centroid, respectively.

Step 4: Re-compute the Centroids Using the Current Cluster Memberships

Person	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Age	19	67	35	60	65	49	70	70	57	68	23	65	27	47	57	43	56	40	37	34
Income $\times 1000$	15	19	24	30	38	42	46	49	54	59	62	63	67	71	75	78	79	87	97	103
Expense score 1-100	39	14	35	4	35	52	56	55	51	55	41	52	56	9	5	17	35	13	32	23
DC1	62	50	46	55	23	22	0	3	16	13	52	18	48	58	60	57	42	67	65	75
DC2	55	75	49	72	52	34	48	47	33	42	16	38	0	51	60	44	38	49	40	49
DC3	84	86	73	76	65	60	65	63	51	54	39	48	40	36	40	25	26	22	0	11
Cluster	2	1	1	1	1	1	1	1	1	1	2	1	2	3	3	3	3	3	3	3

- New 1st centroid:

$$x_1 = (67 + 35 + 60 + 65 + 49 + 70 + 70 + 57 + 68 + 65)/10 = 60.6$$
$$x_2 = (19 + 24 + 30 + 38 + 42 + 46 + 49 + 54 + 59 + 63)/10 = 42.4$$
$$x_3 = (14 + 35 + 4 + 35 + 52 + 56 + 55 + 51 + 55 + 52)/10 = 40.9$$

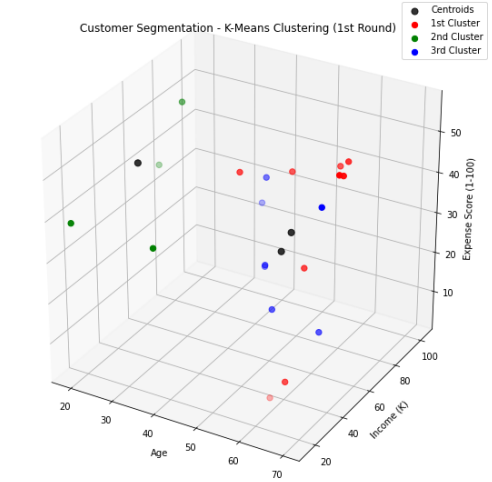
- New 2nd centroid:

$$x_1 = (19 + 23 + 27)/3 = 23$$
$$x_2 = (15 + 62 + 67)/3 = 48$$
$$x_3 = (39 + 41 + 56)/3 = 45.33$$

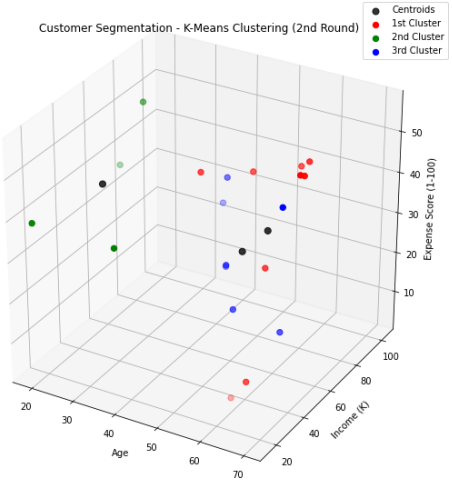
- New 3rd centroid:

$$x_1 = (47 + 57 + 43 + 56 + 40 + 37 + 34)/7 = 44.86$$
$$x_2 = (71 + 75 + 78 + 79 + 87 + 97 + 103)/7 = 84.29$$
$$x_3 = (9 + 5 + 17 + 35 + 13 + 32 + 23)/7 = 19.14$$

The 1st and the 2nd Round

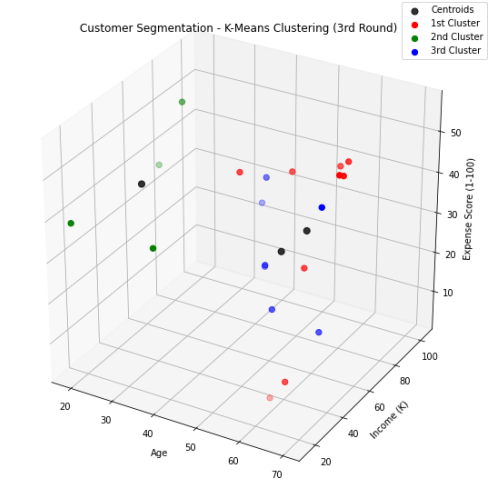


1st centroid: (70, 46, 56)
2nd centroid: (27, 67, 56)
3rd centroid: (37, 97, 32)

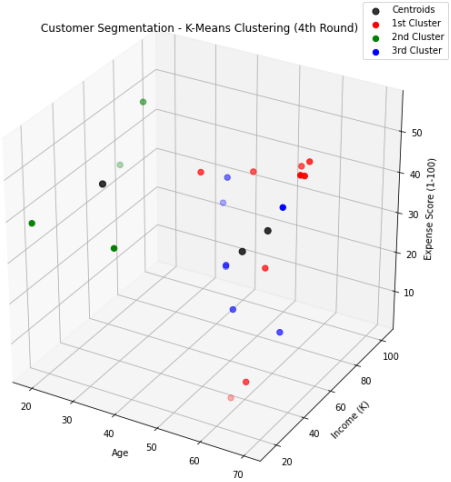


1st centroid: (60.6, 42.4, 40.9)
2nd centroid: (23, 48, 45.3)
3rd centroid: (44.9, 84.3, 19.1)

The 3rd and the 4th Round



1st centroid: (63.4, 44.4, 41.6)
2nd centroid: (26, 42, 42.8)
3rd centroid: (44.9, 84.3, 19.1)



1st centroid: (63.4, 44.4, 41.6)
2nd centroid: (26, 42, 42.8)
3rd centroid: (44.9, 84.3, 19.1)

Conclusion

centroid.

- Cluster 1:
  The average age is 63 years old, the average annual income is \$44K, and the average expense score is 42 of 100.
- Cluster 2:
  The average age is 26 years old, the average annual income is \$42K, and the average expense score is 43 of 100.
- Cluster 3:
  The average age is 45 years old, the average annual income is \$84K, and the average expense score is 19 of 100.

K-Means Clustering Implementation using Scikit-Learn

```
# Import the required libraries
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Unlabled training data
data = np.array([[19, 15, 39], [67, 19, 14], [35, 24, 35], [60, 30, 4], [65, 38, 35],
                 [49, 42, 52], [70, 46, 56], [70, 49, 55], [57, 54, 51], [68, 59, 55],
                 [23, 62, 41], [65, 63, 52], [27, 67, 56], [47, 71, 9],  [57, 75, 5],
                 [43, 78, 17], [56, 79, 35], [40, 87, 13], [37, 97, 32], [34, 103, 23]])

# Initial centroids
init_centroids = np.array([[70, 46, 56], [27, 67, 56], [37, 97, 32]])

# Create a KMeans object by specifying
# - Number of clusters (n_clusters) = 3, initial centroids (init) = init_centroids
# - Number of time the k-means algorithm will be run with different centroid seeds (n_init) = 1
# - Maximum number of iterations of the k-means algorithm for a single run (max_iter) = 4
kmeans = KMeans(n_clusters=3, init=init_centroids, n_init=1, max_iter = 4)
```

K-Means Clustering Implementation using Scikit-Learn

```
kmeans.fit(data)                    # Compute k-means clustering
labels = kmeans.predict(data)       # Predict the closest cluster each sample in data belongs to
centroids = kmeans.cluster_centers_ # Get resulting centroids
fig = plt.figure(figsize = (10,10)) # Figure width = 10 inches, height = 10 inches
ax = fig.gca(projection='3d')       # Defining 3D axes so that we can plot 3D data into it

# Get boolean arrays representing entries with labels = 0, 1, and 2
a = np.array(labels == 0); b = np.array(labels == 1); c = np.array(labels == 2)

# Plot centroids with color = black, size = 50 units, transparency = 20%, and put label "Centroids"
ax.scatter(centroids[:,0], centroids[:,1], centroids[:,2],
           c="black", s=50, alpha=0.8, label="Centroids")
# Plot data in the different clusters (1st in red, 2nd in green, 3rd blue)
ax.scatter(data[a,0], data[a,1], data[a,2], c="red", s=40, label="1st Cluster")
ax.scatter(data[b,0], data[b,1], data[b,2], c="green", s=40, label="2nd Cluster")
ax.scatter(data[c,0], data[c,1], data[c,2], c="blue", s=40, label="3rd Cluster")
ax.legend() # Show legend

ax.set_xlabel("Age")                    # Put x-axis label "Age"
ax.set_ylabel("Income (K)")             # Put y-axis label "Income (K)"
ax.set_zlabel("Expense Score (1-100)") # Put z-axis label "Expense Score (1-100)"
ax.set_title("Customer Segmentation - K-Means Clustering") # Put figure title
```

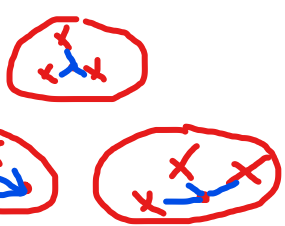
K-Means Stopping Criterion

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

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

where

- C_j is the jth cluster
- $\mathbf{m_j}$ is the centroid of cluster C_j
- dist($\mathbf{x}$, $\mathbf{m_j}$) is the distance between data point $\mathbf{x}$ and centroid $\mathbf{m_j}$



Common Questions

1. How to choose K? (K-means)

   Answer: The following methods are used to find the optimal value of K for K-Means Clustering.

   1. Elbow method ← SSE
   2. ~~Silhouette method~~ ✗ use in this course.

   These methods will be discussed in advanced level machine learning courses.

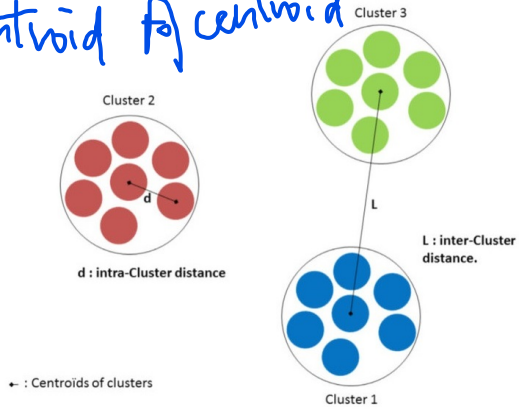
2. What distance metric should be used for K-Means?

   Answer: It depends on your data. Normally, K-Means uses Euclidean distance.

Clustering Quality

- High quality clustering
  1. Maximizes inter-clusters distance (Isolation)
     (i.e., distance between clusters)
  2. Minimizes intra-clusters distance (Compactness)
     (i.e., distance between data points in the same cluster)

centroid to centroid

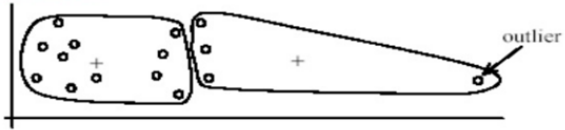


The quality of a clustering result depends on the algorithm, the distance function, and the application.

Weaknesses of K-Means

-ve.

- It is sensitive to outliers
  - Outliers are data points that are very far away from other data points
  - Outliers could be errors in the data recording or some special data points with very different values
  - Desirable and undesirable clustering with outliers



How to Deal with Outliers?

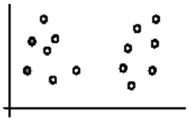
1. Remove some data points in the clustering process that are much further away from the centroids than other data points.
   - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
2. Perform random sampling. Since in sampling, we only choose a small subset of the data points, the chance of select an outlier is very small.
   - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification.

Weaknesses of K-Means

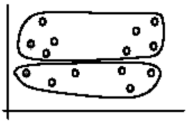
centroids.

- The algorithm is sensitive to initial seeds.

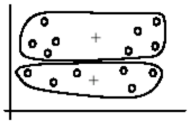
If we use seeds that are not-so-good:
not-so-good results



(A). Random selection of seeds (centroids)



(B). Iteration 1

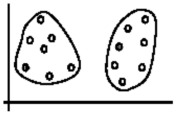


(C). Iteration 2

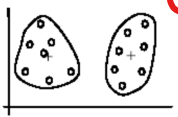
1

If we use different seeds:
good results

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



(B). Iteration 1

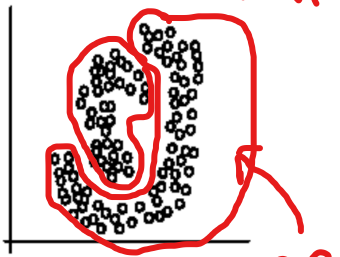


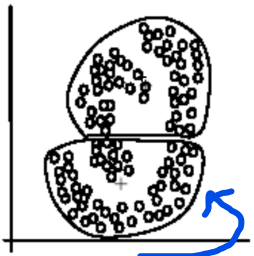
(C). Iteration 2

2

Weaknesses of K-Means

- The K-Means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).





Pros and Cons

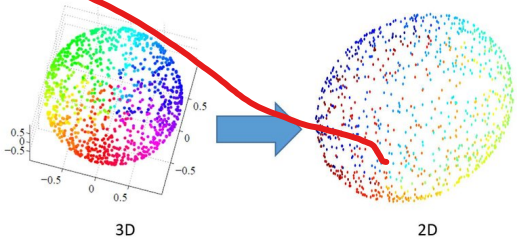
- Pros:
  - Easy to understand and implement
  - It is efficient, given K and the number of iterations is small
- Cons:
  - It is only applicable if the mean is defined.
    For categorical data, K-mode is used, i.e., the centroid is represent by most frequent values
  - The user needs to specify K
  - It is sensitive to outliers

Summary

- Despite weaknesses, K-Means is still the most popular algorithm due to its simplicity, efficiency and other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general, although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters.

Remark: Dimension Reduction using PCA

- Principal Component Analysis (PCA) is one of the easiest, most intuitive and most frequently used methods for dimensionality reduction, projecting data onto its orthogonal feature subspace.
  - The main idea of PCA is to reduce the dimensionality of a data set consisting of many variables correlated with each other, while retaining the variation present in the dataset, up to the maximum extent.
- It is a common practice to apply PCA before K-Means clustering is performed.
- It is believed that it improves the clustering results in practice (noise reduction).



Practice Problem

- Given 4 types of medicines and each has two attributes (weight and pH index).
- Group these medicines into K = 2 group using K-Means clustering.

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4

- Use (1, 1) and (2, 1) as the initial centroids (i.e., seeds) and performing K-Means Clustering until there is no re-assignments of data points to different clusters.

Step 1: Use the 2 Given Data Points as Initial Centroids
Step 2: Find the Distances Between Each Data Point with the 2 Centroids

- 1st centroid: (1, 1)
- 2nd centroid: (2, 1)

Assume Euclidean distance is used.

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0	1	3.6	5
DC2	1	0	2.8	4.2

where DC1, DC2 are the distances between the data points and 1st centroid, and 2nd centroid, respectively.

Step 3: Assign Each Data Point to the Closest Centroid

- 1st centroid: (1, 1)
- 2nd centroid: (2, 1)

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0	1	3.6	5
DC2	1	0	2.8	4.2
Cluster	1	2	2	2

where DC1, DC2 are the distances between the data points and 1st centroid, and 2nd centroid, respectively.

Step 4: Re-compute the Centroids Using the Current Cluster Memberships

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0	1	3.6	5
DC2	1	0	2.8	4.2
Cluster	1	2	2	2

- New 1st centroid:

$$x_1 = 1$$
$$x_2 = 1$$

- New 2nd centroid:

$$x_1 = (2 + 4 + 5)/3 = 3.67$$
$$x_2 = (1 + 3 + 4)/3 = 2.67$$

Step 5: Find the Distance Between Each Data Point with the 2 Centroids

- 1st centroid: (1, 1)
- 2nd centroid: (3.67, 2.67)

Assume Euclidean distance is used.

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0	1	3.6	5
DC2	3.1	2.4	0.5	1.9

where DC1, DC2 are the distances between the data points and 1st centroid, and 2nd centroid, respectively.

Step 6: Assign Each Data Point to the Closest Centroid

- 1st centroid: (1, 1)
- 2nd centroid: (3.67, 2.67)

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0	1	3.6	5
DC2	3.1	2.4	0.5	1.9
Cluster	1	1	2	2

where DC1, DC2 are the distances between the data points and 1st centroid, and 2nd centroid, respectively.

Step 7: Re-compute the Centroids Using the Current Cluster Memberships

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0	1	3.6	5
DC2	3.1	2.4	0.5	1.9
Cluster	1	1	2	2

- New 1st centroid:

$$x_1 = (1 + 2)/2 = 1.5$$
$$x_2 = (1 + 1)/2 = 1$$

- New 2nd centroid:

$$x_1 = (4 + 5)/2 = 4.5$$
$$x_2 = (3 + 4)/2 = 3.5$$

Step 8: Find the Distance Between Each Data Point with the 2 Centroids

- 1st centroid: (1.5, 1)
- 2nd centroid: (4.5, 3.5)

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0.5	0.5	3.2	4.6
DC2	4.3	2.4	0.5	1.9

where DC1, DC2 are the distances between the data points and 1st centroid, and 2nd centroid, respectively.

Step 9: Assign Each Data Point to the Closest Centroid

- 1st centroid: (1.5, 1)
- 2nd centroid: (4.5, 3.5)

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0.5	0.5	3.2	4.6
DC2	4.3	2.4	0.5	1.9
Cluster	1	1	2	2

where DC1, DC2 are the distances between the data points and 1st centroid, and 2nd centroid, respectively.

Step 7: Re-compute the Centroids Using the Current Cluster Memberships

Medicine	A	B	C	D
Weight Index	1	2	4	5
pH Index	1	1	3	4
DC1	0	1	3.6	5
DC2	3.1	2.4	0.5	1.9
Cluster	1	1	2	2

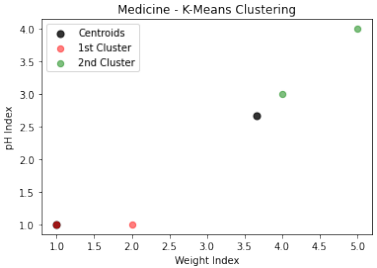
- New 1st centroid:

$$x_1 = (1 + 2)/2 = 1.5$$
$$x_2 = (1 + 1)/2 = 1$$

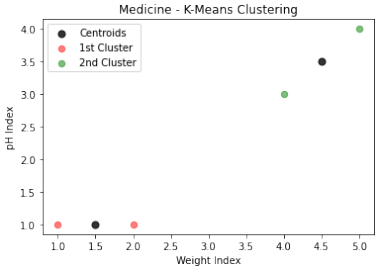
- New 2nd centroid:

$$x_1 = (4 + 5)/2 = 4.5$$
$$x_2 = (3 + 4)/2 = 3.5$$

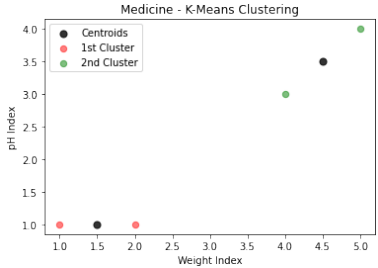
The 1st, the 2nd, and the 3rd Round



1st centroid: (1, 1)
2nd centroid: (3.67, 2.67)



1st centroid: (1.5, 1)
2nd centroid: (4.5, 3.5)



1st centroid: (1.5, 1)
2nd centroid: (4.5, 3.5)

Conclusion

- Cluster 1:
  The average weight index is 1.5, and the average pH index is 1.
- Cluster 2:
  The average weight index is 4.5, and the average pH index is 3.5.

K-Means Clustering Implementation using Scikit-Learn

```
# Import the required libraries
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Unlabled training data
data = np.array([[1,1], [2,1], [4,3], [5,4]])

# Initial centroids
init_centroids = np.array([[1,1], [2,1]])

# Create a KMeans object by specifying
# - Number of clusters (n_clusters) = 2, initial centroids (init) = init_centroids
# - Number of time the k-means algorithm will be run with different centroid seeds (n_init) = 1
# - Maximum number of iterations of the k-means algorithm for a single run (max_iter) = 3
kmeans = KMeans(n_clusters=2, init=init_centroids, n_init=1, max_iter = 3)
```

K-Means Clustering Implementation using Scikit-Learn

```
kmeans.fit(data)                    # Compute k-means clustering
labels = kmeans.predict(data)       # Predict the closest cluster each sample in data belongs to
centroids = kmeans.cluster_centers_ # Get resulting centroids
fig, ax = plt.subplots()            # Defining 2D axes so that we can plot 2D data into it

# Get boolean arrays representing entries with labels = 0 and 1
a = np.array(labels == 0); b = np.array(labels == 1)

# Plot centroids with color = black, size = 50 units, transparency = 20%, and put label "Centroids"
ax.scatter(centroids[:,0], centroids[:,1], c="black", s=50, alpha=0.8, label="Centroids")
# Plot data in the different clusters (1st in red, 2nd in green) with transparency = 50%
ax.scatter(data[a,0], data[a,1], c="red", s=40, alpha=0.5, label="1st Cluster")
ax.scatter(data[b,0], data[b,1], c="green", s=40, alpha=0.5, label="2nd Cluster")
ax.legend() # Show legend

ax.set_xlabel("Weight Index")                 # Put x-axis label "Weight Index"
ax.set_ylabel("pH Index")                     # Put y-axis label "pH Index"
ax.set_title("Medicine - K-Means Clustering") # Put figure title
```

That's all!

Any questions?