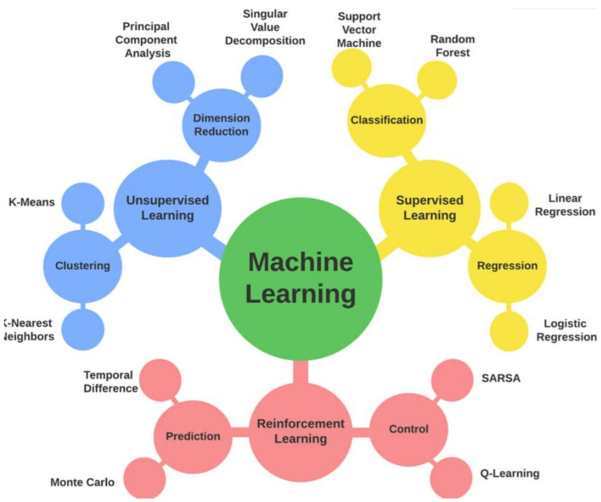
Các phương pháp học máy
Machine learning methods

4 TC: 2 LT – 2 TH

Giảng viên: **Tạ Hoàng Thắng**

tahoangthang@gmail.com

0975399307



Cluster Analysis

Definitions

Wikipedia: **Cluster analysis or clustering** is the task of **grouping a set of objects in such a way that objects in the same group** (called a cluster) are **more similar** (in some specific sense defined by the analyst) **to each other than to those in other groups** (clusters).
https://en.wikipedia.org/wiki/Cluster_analysis

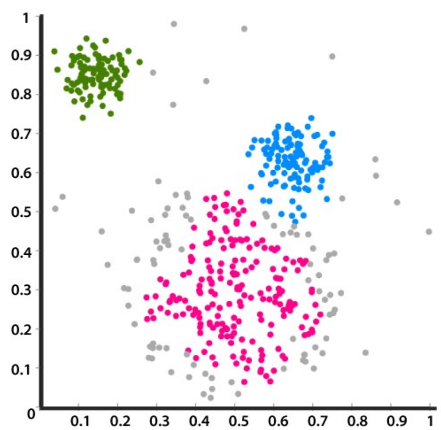
- It is a main task of exploratory data analysis, and a common technique for statistical data analysis,
  - used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

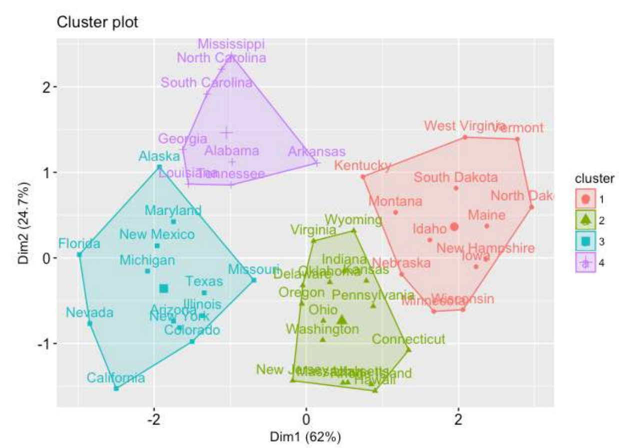
Cluster Analysis

Definitions

ChatGPT: **Clustering analysis** is a **type of unsupervised learning technique** used in data analysis **to group similar data points into clusters** or groups.

- The primary goal is to identify natural patterns or structures in the data by finding groups where data points within the same group (or cluster) are more similar to each other than to those in different groups.





Cluster Analysis

Input:

- **Data points**: The input is a dataset consisting of multiple data points (observations or samples). Each data point is usually represented as a vector of features (attributes). The dataset could be in different forms, such as:
  - **Numeric data**: Features represented by continuous or discrete numerical values (e.g., customer age, income)
  - **Categorical data**: Features represented by categories or labels (e.g., product type, customer gender).
  - **Mixed data**: A combination of numerical and categorical features.

Cluster Analysis

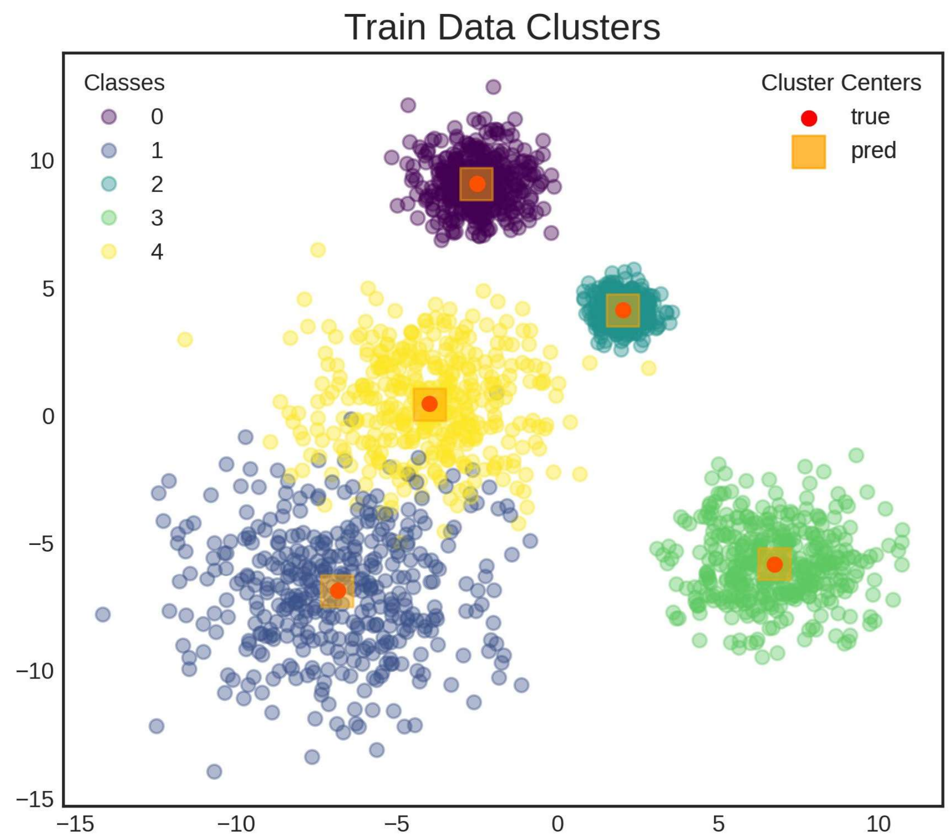
Input:

- **Parameters**:
  - **Number of clusters (K)**: For algorithms like K-Means, the desired number of clusters must be predefined.
  - **Distance metric**: How similarity or distance between data points is measured (e.g., Euclidean distance, Manhattan distance).
  - **Other hyperparameters**: Such as the minimum number of points to form a dense region in DBSCAN, or initial means for K-Means.

Cluster Analysis

Output

- **Cluster assignments**: A label or identifier for each data point indicating which cluster it belongs to.
- **Cluster centers (Centroids)**: For algorithms like K-Means, the output includes the coordinates of the cluster centers or centroids.
- **Cluster sizes**: The number of data points in each cluster.



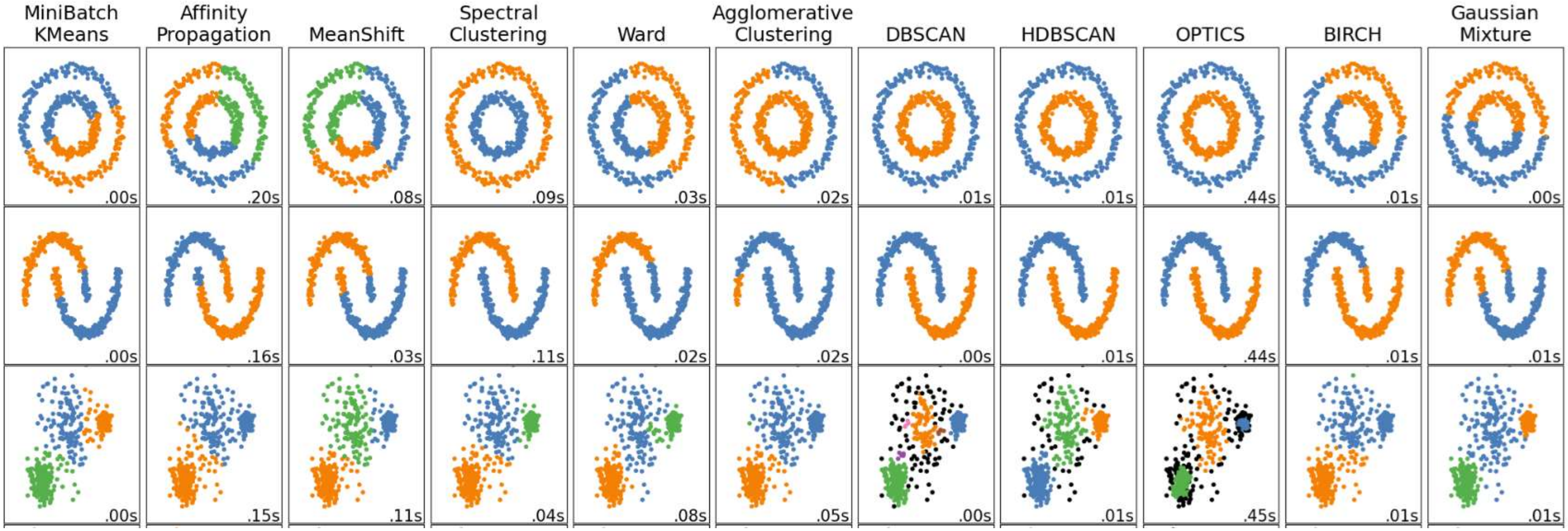
Cluster Analysis

Output

- **Silhouette score or Other metrics**: Evaluation metrics that indicate how well the data has been clustered.
  - A higher silhouette score, for example, means that the data points are well matched to their own cluster and poorly matched to neighboring clusters.
- **Cluster structure**: For hierarchical clustering, the output may include a dendrogram, which visually represents the merging or splitting of clusters at various levels.

Cluster Analysis

Methods



Cluster Analysis

Methods

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points

Cluster Analysis

Methods

Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes, outlier removal, transductive	Distances between nearest points

Cluster Analysis

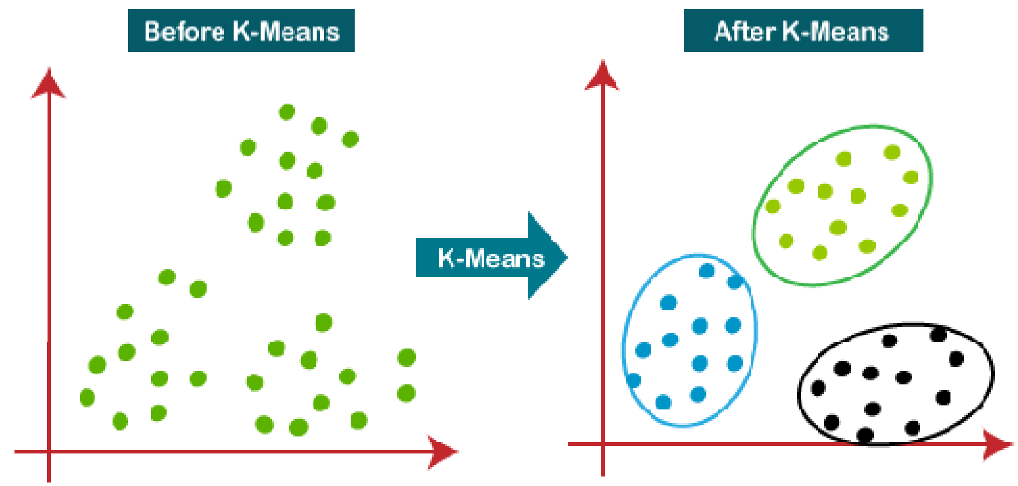
Methods

HDBSCAN	minimum cluster membership, minimum point neighbors	large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes, outlier removal, transductive, hierarchical, variable cluster density	Distances between nearest points
OPTICS	minimum cluster membership	Very large n_samples, large n_clusters	Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points

K-means

K-Means

- is a popular clustering algorithm used in machine learning and data analysis to partition a dataset into a specified number of clusters (**K**).
- the goal of K-Means is to divide the data points into **K** clusters
  - each data point belongs to the cluster with the nearest mean (centroid), minimizing the overall variance within each cluster.



K-means

Mathematical Formulas

- Data Points: $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector in a d-dimensional space.
- Number of Clusters: K
- Centroids: $\{\mu_1, \mu_2, \ldots, \mu_K\}$, where each $\mu_k \in \mathbb{R}^d$ represents the center of cluster k.

K-means

Mathematical Formulas

Objective Function (Cost Function)

The K-Means algorithm aims to minimize the **within-cluster sum of squares (WCSS)**, also called the inertia:

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \mu_k\|^2$$

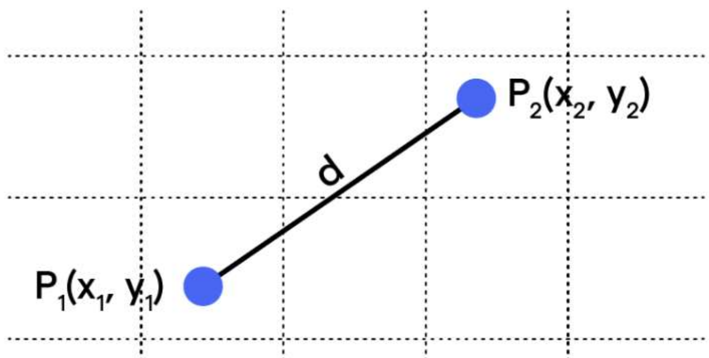
where:

- C_k is the set of points assigned to cluster k.
- $\|\mathbf{x}_i - \mu_k\|^2$ is the squared Euclidean distance between a data point $\mathbf{x}_i$ and the centroid μ_k.

K-means

Mathematical Formulas

- **Group Exercise:**
  - Calculate the distance between **x1(5,6,7)** and **x2(8,9,100)** using Euclidean distance?



$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{i=1}^{d} (x_{1i} - x_{2i})^2}$$

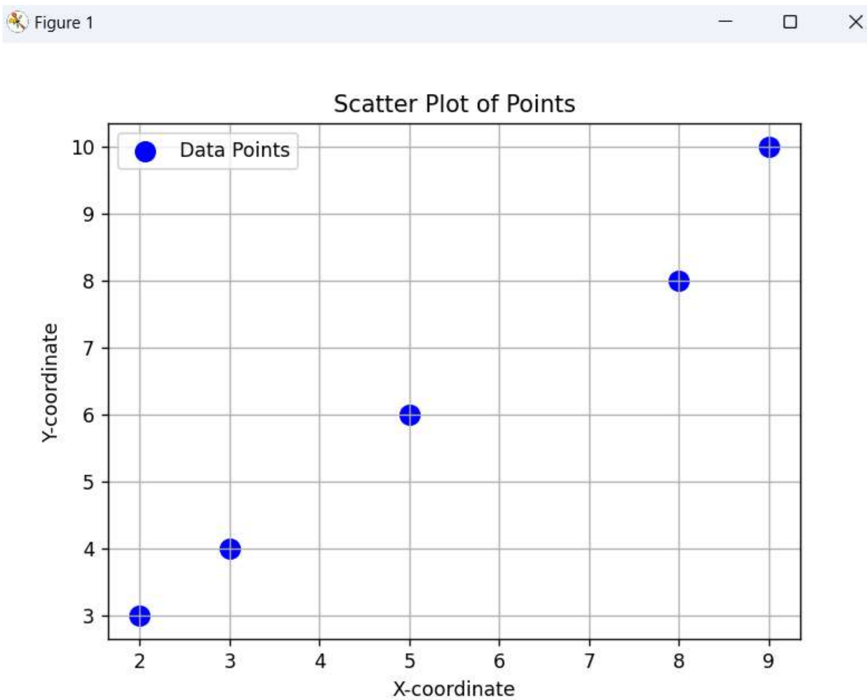
Euclidean Distance (d) $= \sqrt{(x_2 - y_1)^2 + (y_2 - y_1)^2}$

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

```
import matplotlib.pyplot as plt

# Dataset
X = [(2, 3), (3, 4), (5, 6), (8, 8), (9, 10)]

# Extract X and Y coordinates
x_coords = [point[0] for point in X]
y_coords = [point[1] for point in X]

# Create a scatter plot
plt.scatter(x_coords, y_coords, color='blue', marker='o',
            s=100, label='Data Points')

# Add titles and labels
plt.title('Scatter Plot of Points')
plt.xlabel('X-coordinate')
plt.ylabel('Y-coordinate')
plt.grid(True)
plt.legend()

# Display the plot
plt.show()
```



K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Step 1: Initialization

1. Choose the number of clusters, $K = 2$.
2. Randomly initialize the centroids.

   Let's randomly select two initial centroids from the data points:

$$\mu_1 = (2,3), \quad \mu_2 = (8,8)$$

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Step 2: Assignment Step

Assign each point to the nearest centroid using the Euclidean distance.

The distance between a point (x_1, y_1) and a centroid (x_2, y_2) is calculated as:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Point	Distance to $\mu_1 = (2,3)$	Distance to $\mu_2 = (8,8)$	Nearest Centroid
(2, 3)	$\sqrt{(2-2)^2+(3-3)^2} = 0$	$\sqrt{(2-8)^2+(3-8)^2} = \sqrt{61} \approx 7.81$	μ_1
(3, 4)	$\sqrt{(3-2)^2+(4-3)^2} = \sqrt{2} \approx 1.41$	$\sqrt{(3-8)^2+(4-8)^2} = \sqrt{41} \approx 6.40$	μ_1
(5, 6)	$\sqrt{(5-2)^2+(6-3)^2} = \sqrt{18} \approx 4.24$	$\sqrt{(5-8)^2+(6-8)^2} = \sqrt{13} \approx 3.61$	μ_2
(8, 8)	$\sqrt{(8-2)^2+(8-3)^2} = \sqrt{61} \approx 7.81$	$\sqrt{(8-8)^2+(8-8)^2} = 0$	μ_2
(9, 10)	$\sqrt{(9-2)^2+(10-3)^2} = \sqrt{98} \approx 9.90$	$\sqrt{(9-8)^2+(10-8)^2} = \sqrt{5} \approx 2.24$	μ_2

Cluster Assignments:

- Cluster 1 (μ_1): $(2,3), (3,4)$
- Cluster 2 (μ_2): $(5,6), (8,8), (9,10)$

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Cluster Assignments:

- Cluster 1 (μ_1): $(2, 3), (3, 4)$
- Cluster 2 (μ_2): $(5, 6), (8, 8), (9, 10)$

Step 3: Update Step

Calculate the new centroids for each cluster by finding the mean of the points assigned to each cluster.

1. **New Centroid for Cluster 1 (μ_1):**

$$\mu_1 = \left(\frac{2+3}{2}, \frac{3+4}{2}\right) = (2.5, 3.5)$$

2. **New Centroid for Cluster 2 (μ_2):**

$$\mu_2 = \left(\frac{5+8+9}{3}, \frac{6+8+10}{3}\right) = \left(\frac{22}{3}, \frac{24}{3}\right) \approx (7.33, 8.0)$$

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Step 4: Repeat Assignment Step

Reassign each point to the nearest updated centroid:

Point	Distance to $\mu_1 = (2.5, 3.5)$	Distance to $\mu_2 = (7.33, 8.0)$	Nearest Centroid
(2, 3)	$\sqrt{(2-2.5)^2+(3-3.5)^2} = \sqrt{0.5} \approx 0.71$	$\sqrt{(2-7.33)^2+(3-8)^2} \approx 7.45$	μ_1
(3, 4)	$\sqrt{(3-2.5)^2+(4-3.5)^2} = \sqrt{0.5} \approx 0.71$	$\sqrt{(3-7.33)^2+(4-8)^2} \approx 6.02$	μ_1
(5, 6)	$\sqrt{(5-2.5)^2+(6-3.5)^2} = \sqrt{12.5} \approx 3.54$	$\sqrt{(5-7.33)^2+(6-8)^2} \approx 3.14$	μ_2
(8, 8)	$\sqrt{(8-2.5)^2+(8-3.5)^2} = \sqrt{60.5} \approx 7.78$	$\sqrt{(8-7.33)^2+(8-8)^2} = 0.67$	μ_2
(9, 10)	$\sqrt{(9-2.5)^2+(10-3.5)^2} = \sqrt{98} \approx 9.90$	$\sqrt{(9-7.33)^2+(10-8)^2} = 2.69$	μ_2

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Step 4: Repeat Assignment Step

Reassign each point to the nearest updated centroid:

Point	Distance to $\mu_1 = (2.5, 3.5)$	Distance to $\mu_2 = (7.33, 8.0)$	Nearest Centroid
(2, 3)	$\sqrt{(2-2.5)^2+(3-3.5)^2} = \sqrt{0.5} \approx 0.71$	$\sqrt{(2-7.33)^2+(3-8)^2} \approx 7.45$	μ_1
(3, 4)	$\sqrt{(3-2.5)^2+(4-3.5)^2} = \sqrt{0.5} \approx 0.71$	$\sqrt{(3-7.33)^2+(4-8)^2} \approx 6.02$	μ_1
(5, 6)	$\sqrt{(5-2.5)^2+(6-3.5)^2} = \sqrt{12.5} \approx 3.54$	$\sqrt{(5-7.33)^2+(6-8)^2} \approx 3.14$	μ_2
(8, 8)	$\sqrt{(8-2.5)^2+(8-3.5)^2} = \sqrt{60.5} \approx 7.78$	$\sqrt{(8-7.33)^2+(8-8)^2} = 0.67$	μ_2
(9, 10)	$\sqrt{(9-2.5)^2+(10-3.5)^2} = \sqrt{98} \approx 9.90$	$\sqrt{(9-7.33)^2+(10-8)^2} = 2.69$	μ_2

New Cluster Assignments:

- Cluster 1 (μ_1): $(2,3), (3,4)$
- Cluster 2 (μ_2): $(5,6), (8,8), (9,10)$

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

```python
import numpy as np
import matplotlib.pyplot as plt

# Dataset
X = np.array([[2, 3], [3, 4], [5, 6], [8, 8], [9, 10]])

# Number of clusters
K = 2

# Randomly initialize centroids (using two points from the dataset as initial centroids)
centroids = np.array([[2, 3], [8, 8]])

# Function to compute Euclidean distance
def euclidean_distance(a, b):
    return np.sqrt(np.sum((a - b) ** 2))

# Function to assign each point to the nearest centroid
def assign_clusters(X, centroids):
    clusters = []
    for point in X:
        distances = [euclidean_distance(point, centroid) for centroid in centroids]
        cluster = np.argmin(distances)
        clusters.append(cluster)
    return np.array(clusters)

# Function to update centroids by calculating the mean of the points in each cluster
def update_centroids(X, clusters, K):
    new_centroids = []
    for k in range(K):
        cluster_points = X[clusters == k]
        new_centroid = np.mean(cluster_points, axis=0)
        new_centroids.append(new_centroid)
    return np.array(new_centroids)
```

K-means

Simple example: Manual Calculation

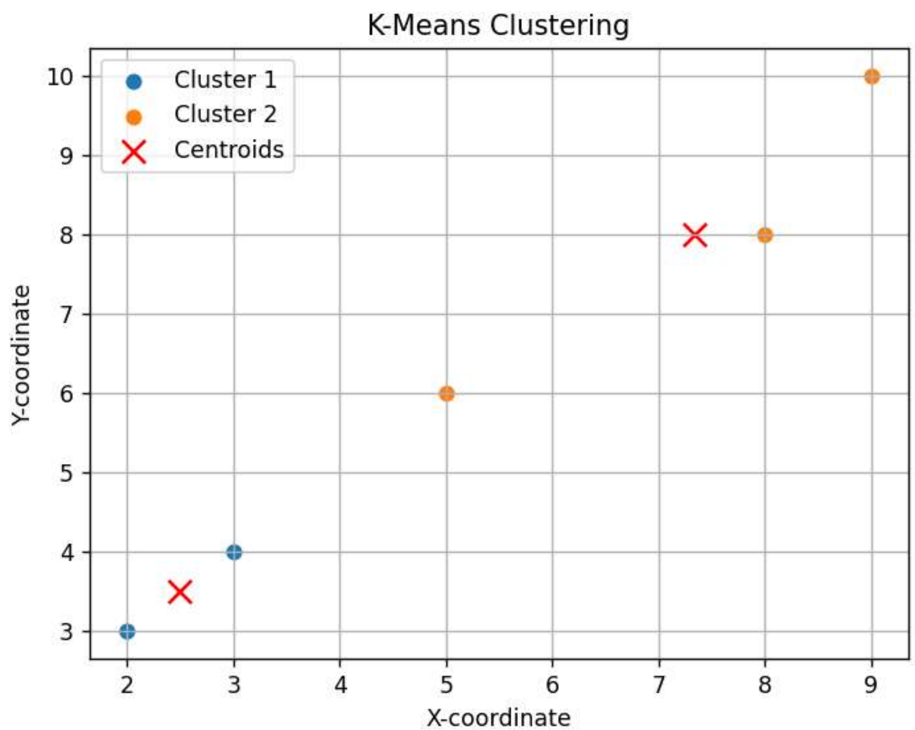
X={(2,3),(3,4),(5,6),(8,8),(9,10)}

```
# K-Means Clustering Algorithm
max_iterations = 10
for iteration in range(max_iterations):
    # Step 1: Assign clusters
    clusters = assign_clusters(X, centroids)

    # Step 2: Update centroids
    new_centroids = update_centroids(X, clusters, K)

    # Check for convergence (if centroids do not change)
    if np.all(centroids == new_centroids):
        break
    centroids = new_centroids

# Print final centroids and clusters
print("Final centroids:")
print(centroids)
print("Cluster assignments:")
print(clusters)

# Visualize the clusters
for k in range(K):
    cluster_points = X[clusters == k]
    plt.scatter(cluster_points[:, 0], cluster_points[:, 1], label=f'Cluster {k+1}')
plt.scatter(centroids[:, 0], centroids[:, 1], color='red', marker='x', s=100, label='Centroids')
plt.xlabel('X-coordinate')
plt.ylabel('Y-coordinate')
plt.title('K-Means Clustering')
plt.legend()
plt.grid(True)
plt.show()
```

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

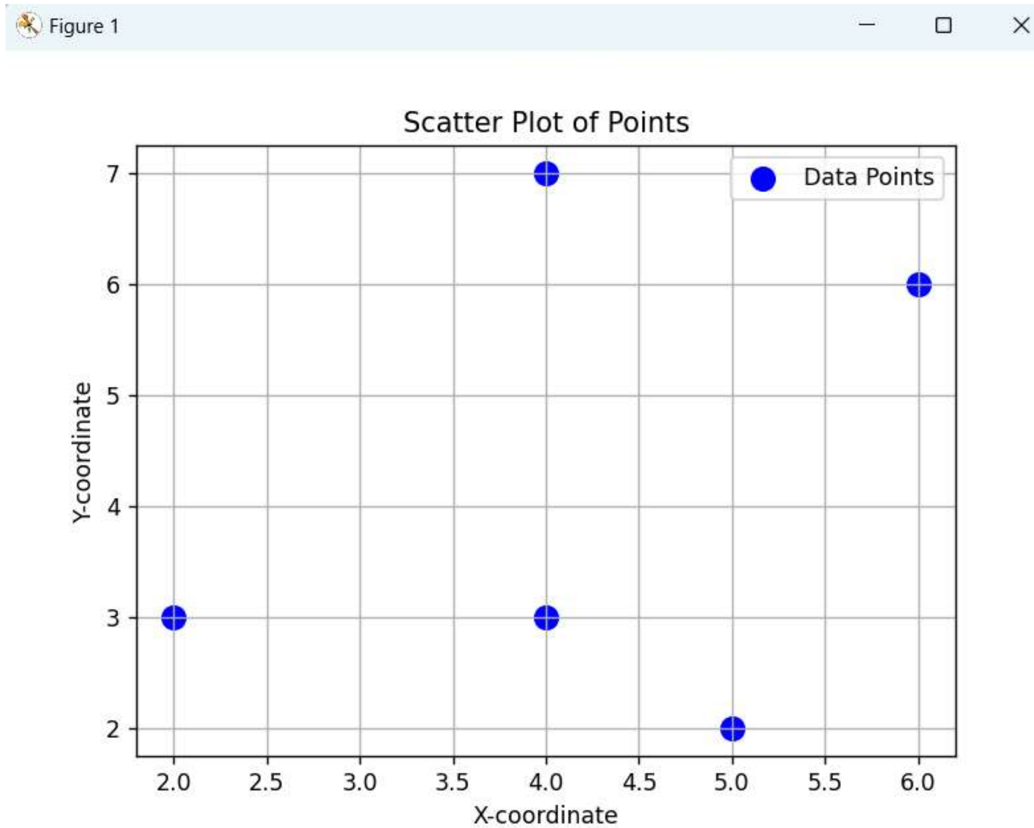


K-means

Simple example: Manual Calculation

Group Excersise:

- Repeat the manual calculation with: **X = [(2,3),(4,3),(5,2),(6,6),(4,7)]**
- Choose the **μ1=(2,3), μ2=(6,6)**

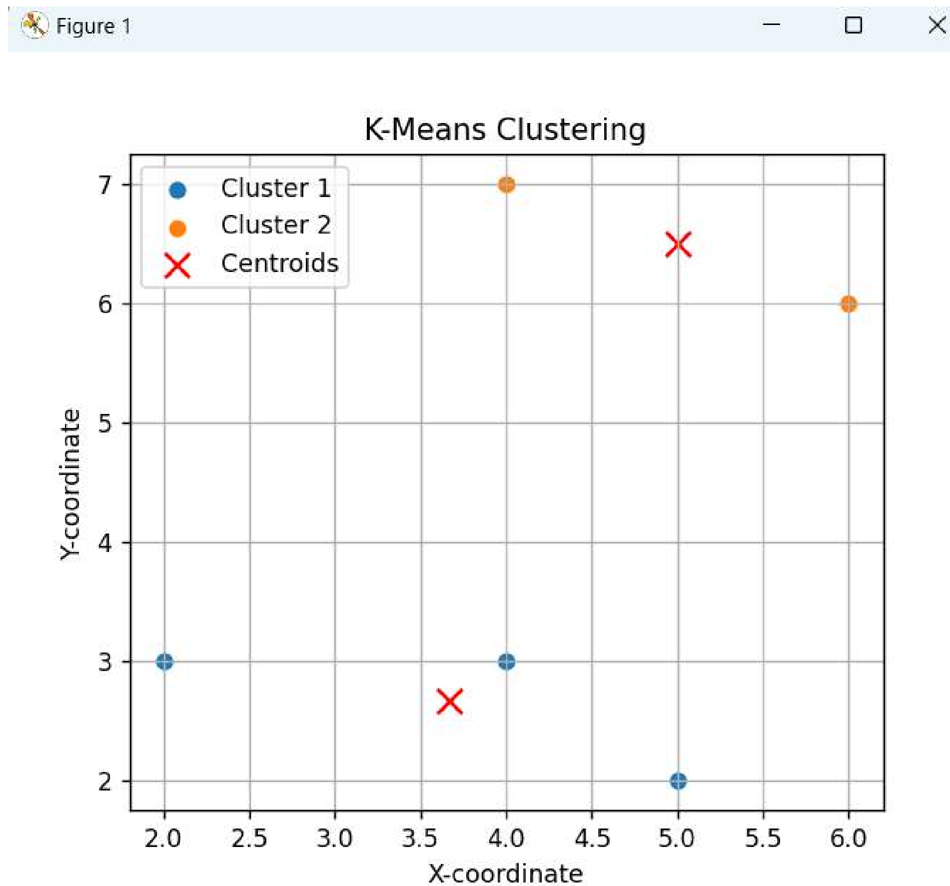


K-means

Simple example: Manual Calculation

Group Excersise:

- Repeat the manual calculation with: **X = [(2,3),(4,3),(5,2),(6,6),(4,7)]**
- Choose the **μ1=(2,3), μ2=(6,6)**

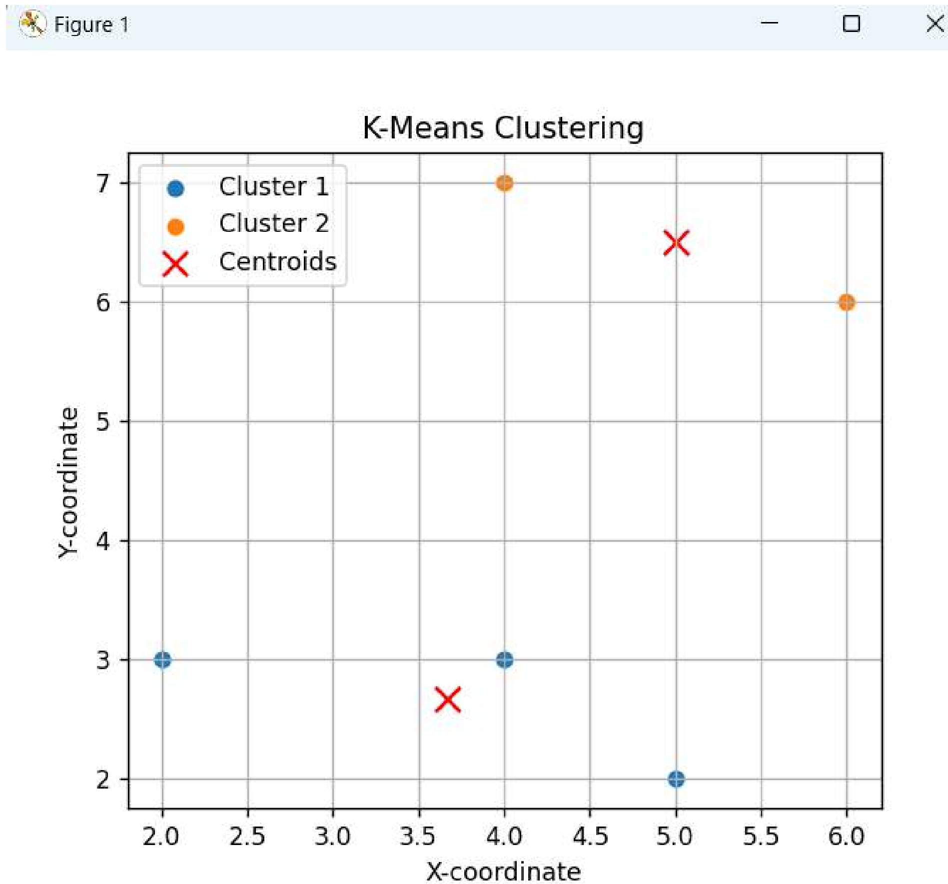


K-means

Simple example: Manual Calculation

Group Excersise:

- Repeat the manual calculation with: **X = [(2,3),(4,3),(5,2),(6,6),(4,7)]**
- Choose the **µ1=(2,3), µ2=(6,6)**



K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Python Code Using Scikit-Learn:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Dataset
X = np.array([[2, 3], [3, 4], [5, 6], [8, 8], [9, 10]])

# Apply K-Means clustering
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)

# Get cluster assignments and centroids
y_kmeans = kmeans.predict(X)
centroids = kmeans.cluster_centers_
```

K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Python Code Using Scikit-Learn:

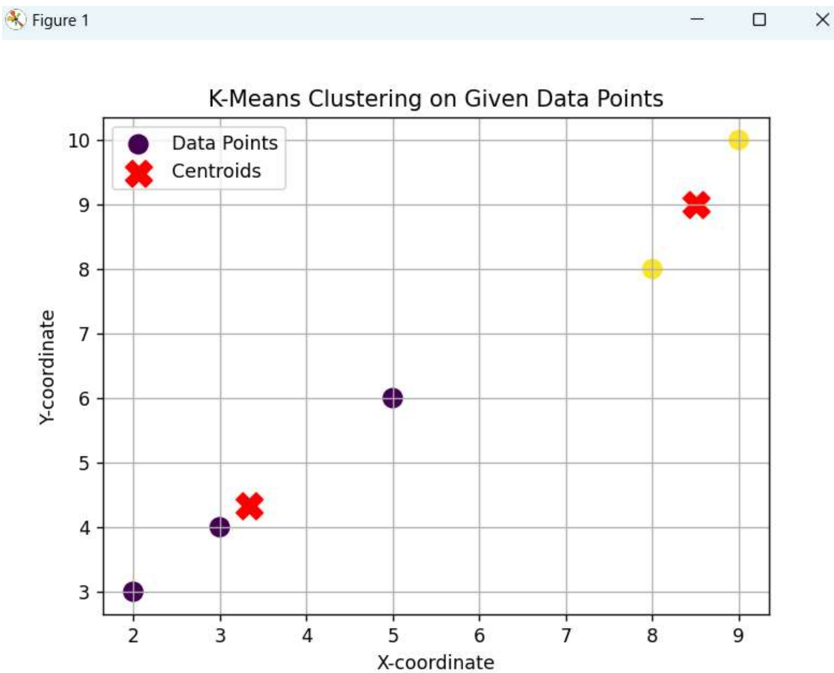
```python
# Plot the data points
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=100,
            cmap='viridis', label='Data Points')

# Plot the centroids
plt.scatter(centroids[:, 0], centroids[:, 1], s=200,
            c='red', marker='X', label='Centroids')

# Add titles and labels
plt.title('K-Means Clustering on Given Data Points')
plt.xlabel('X-coordinate')
plt.ylabel('Y-coordinate')
plt.legend()
plt.grid(True)

# Show the plot
plt.show()
```

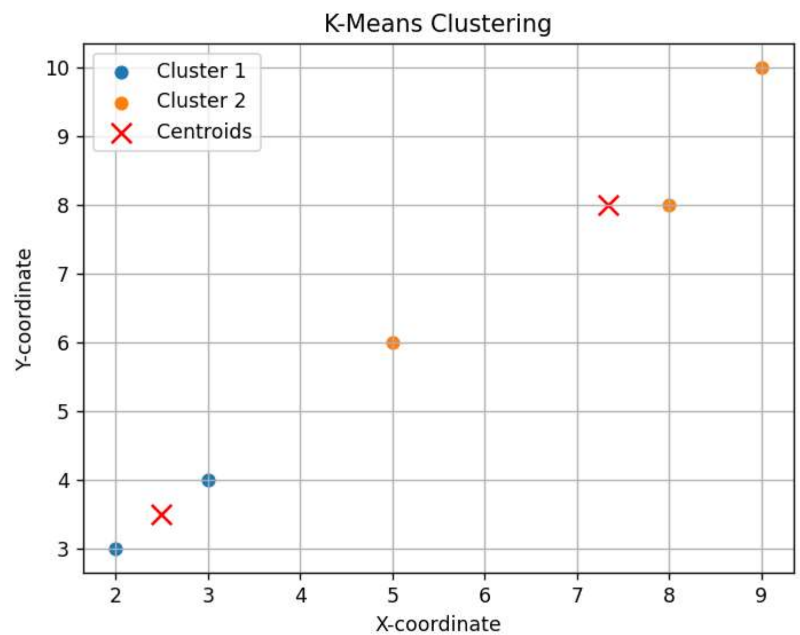
K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Python Code Using Scikit-Learn:

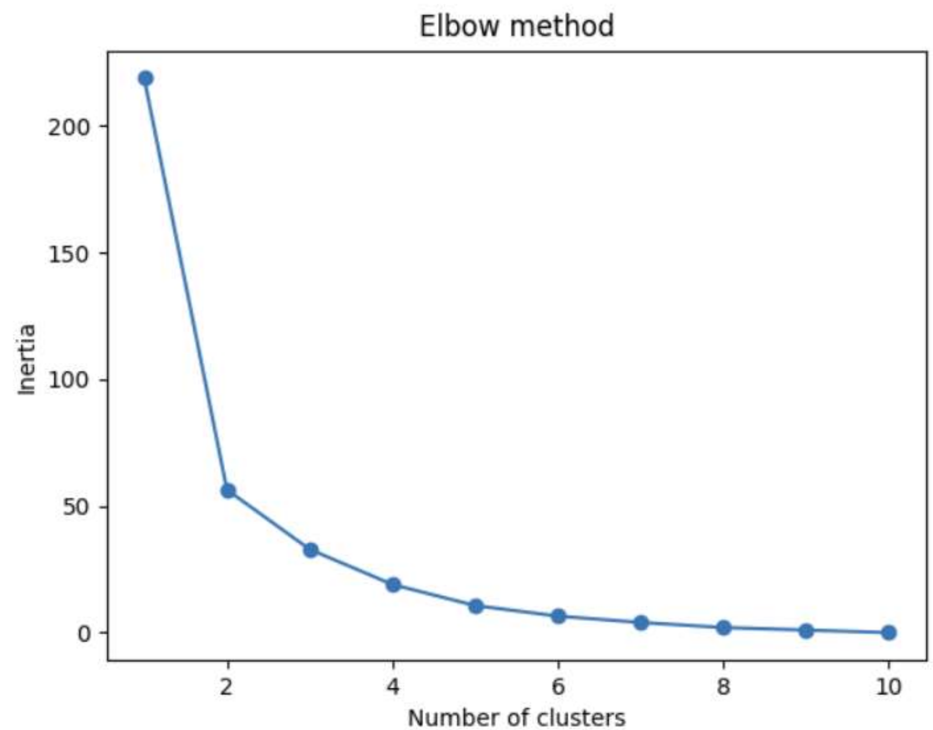




K-means

The elbow method

- is a technique used to determine the optimal number of clusters k for K-Means clustering.
- involves **plotting the sum of squared distances from each point to its assigned cluster centroid (inertia)** as a function of k and **looking for a point where the rate of decrease sharply slows down**
  - this point is referred to as the "elbow”.

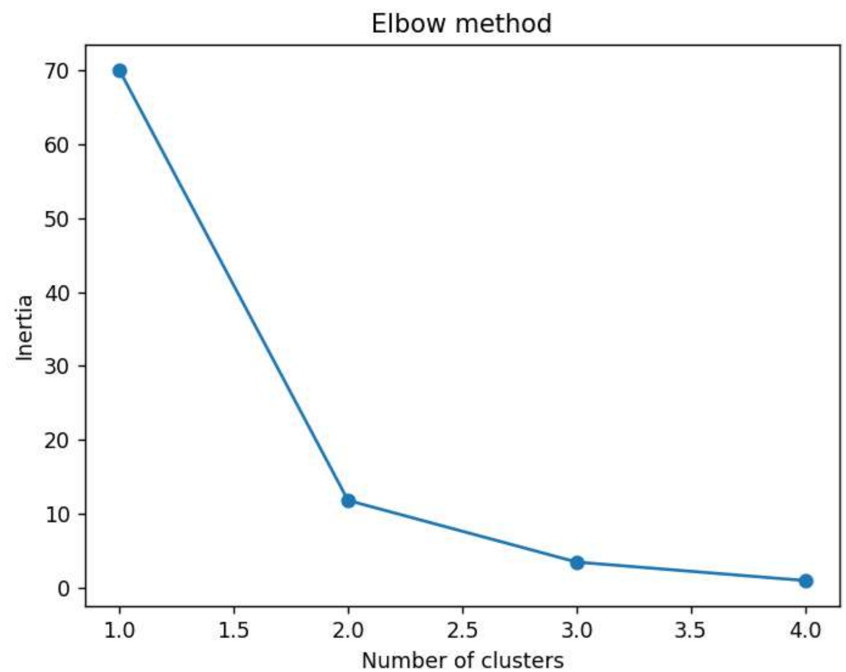


K-means

The elbow method

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

x = [2, 3, 5, 8, 9]
y = [3, 4, 6, 8, 10]
data = list(zip(x, y))
inertias = []

for i in range(1,5):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)

plt.plot(range(1,5), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```



K-means

Simple example: Manual Calculation

X={(2,3),(3,4),(5,6),(8,8),(9,10)}

Exercise:

- **Calculate Inertia 1 and Inertia 2, and their sum?**

Cluster Reassignments:

- Cluster 1: $(2, 3), (3, 4), (5, 6)$
- Cluster 2: $(8, 8), (9, 10)$

$$\text{Inertia} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Centroid for Cluster 1:

$$\mu_1 = \left(\frac{2+3+5}{3}, \frac{3+4+6}{3}\right) = \left(\frac{10}{3}, \frac{13}{3}\right) \approx (3.67, 4.33)$$

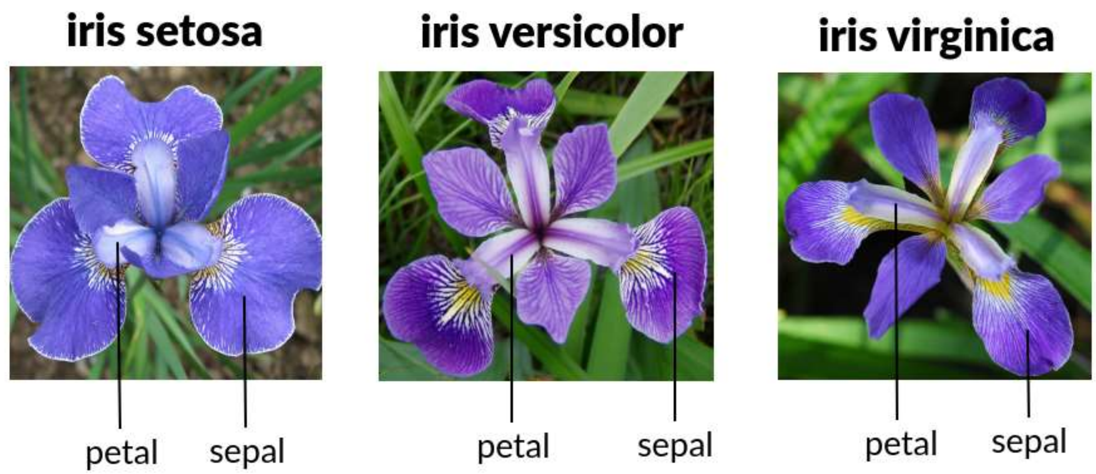
Centroid for Cluster 2:

$$\mu_2 = \left(\frac{8+9}{2}, \frac{8+10}{2}\right) = (8.5, 9)$$

K-means

Toy example

Iris dataset




```
#Iris dataset with labels:
#    sepal length (cm)  sepal width (cm)  ...  petal width (cm)  species
# 0                5.1               3.5  ...               0.2   setosa
# 1                4.9               3.0  ...               0.2   setosa
# 2                4.7               3.2  ...               0.2   setosa
# 3                4.6               3.1  ...               0.2   setosa
# 4                5.0               3.6  ...               0.2   setosa
# [5 rows x 5 columns]
```

K-means

Toy example

Iris dataset

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)  # We know there are 3 species of iris
kmeans.fit(X_scaled)
y_kmeans = kmeans.predict(X_scaled)
centroids = kmeans.cluster_centers_
```

K-means

Toy example

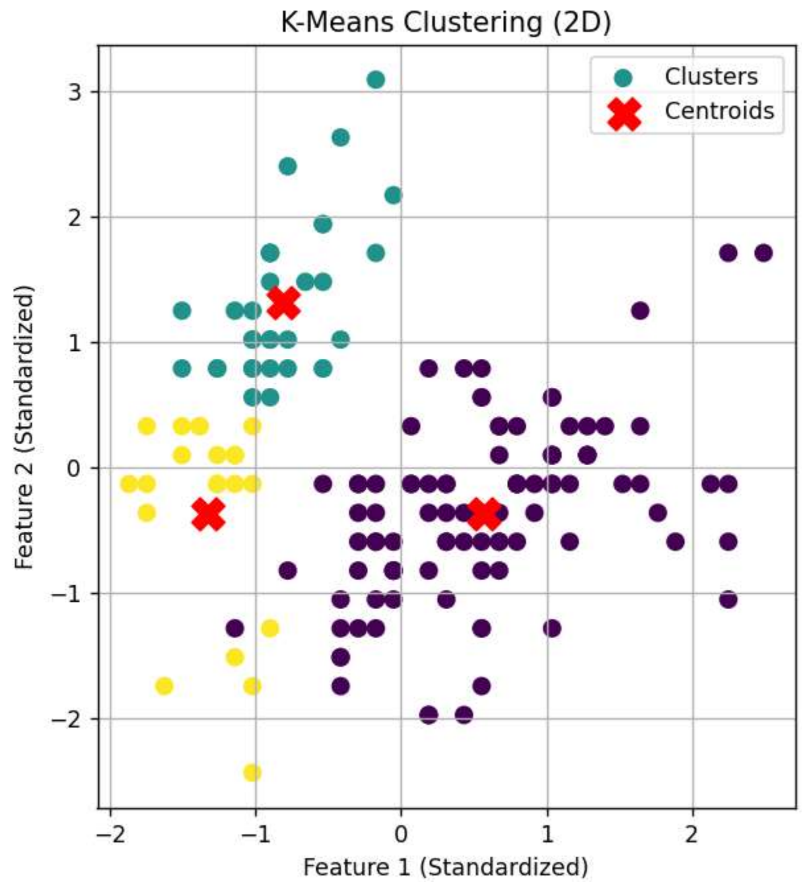
Iris dataset

```
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)  # We know there are 3 species of iris
kmeans.fit(X_scaled)
y_kmeans = kmeans.predict(X_scaled)
centroids = kmeans.cluster_centers_

# 2D Plot (using only the first two features for simplicity)
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1],
            c=y_kmeans, s=50, cmap='viridis', label='Clusters')
plt.scatter(centroids[:, 0], centroids[:, 1],
            s=200, c='red', marker='X', label='Centroids')
plt.title('K-Means Clustering (2D)')
plt.xlabel('Feature 1 (Standardized)')
plt.ylabel('Feature 2 (Standardized)')
plt.legend()
plt.grid(True)
plt.show()
```

K-means

Toy example

Iris dataset



K-means

Advantages of K-Means

- Simplicity
- Scalability: **Handles large datasets well**
- Convergence: Guaranteed convergence
- Speed: Fast execution
- …

Disadvantages of K-Means

- Requires to define the number of clusters
- Sensitive to initial centroids
- **…**