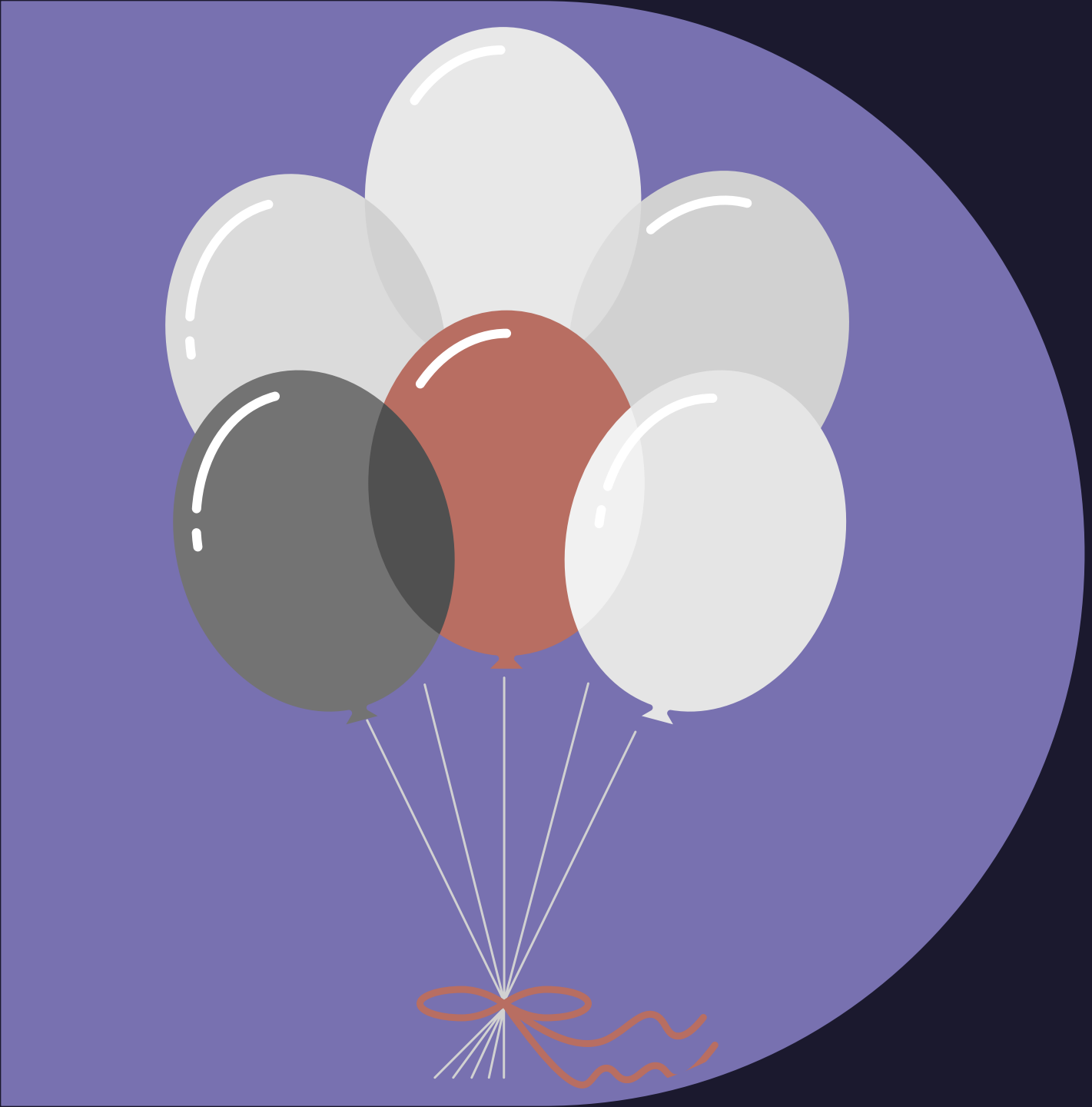
IEEE FCI - Helwan Student Branch



Unsupervised Learning

Hossam Ahmed

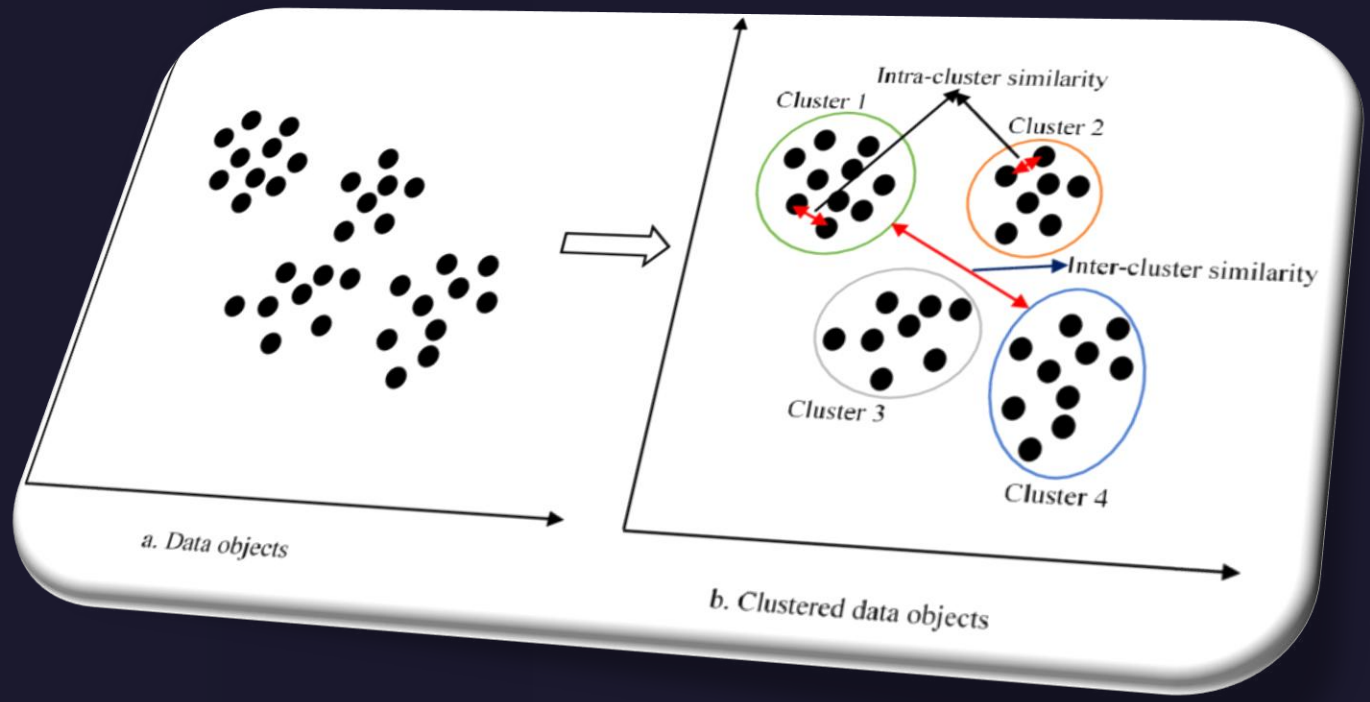
Mario Mamdouh

Unsupervised tasks

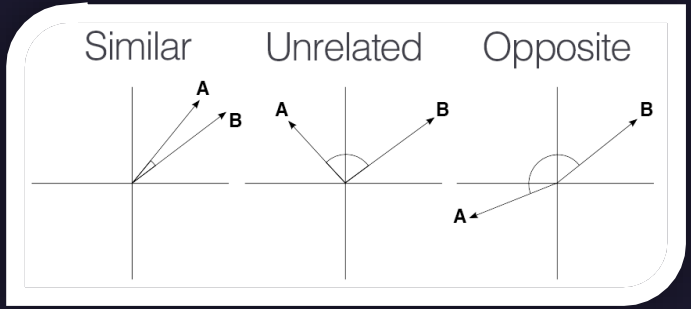
- Clustering: Involves grouping similar data points together based on certain features or characteristics. The goal is to discover natural groupings or clusters within the data.
- Dimensionality Reduction: This task aims to reduce the number of input features while retaining as much relevant information as possible.
- Anomaly Detection: The algorithm identifies data points that deviate significantly from the norm. It is used to detect outliers or anomalies in a dataset, which can be useful for fraud detection, fault detection, and more.
- Density Estimation: The algorithm models the underlying probability distribution of the data. This can be useful for understanding the data's characteristics and generating new samples that follow the learned distribution
- Generative Modeling: Generative models learn to generate new data samples that are similar to the training data

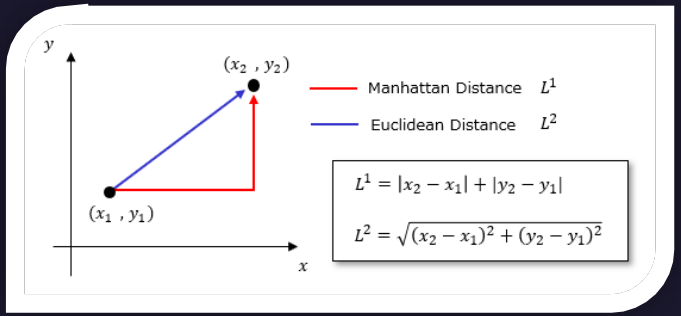
Clustering

- In clustering, the algorithm tries to partition the data into meaningful clusters based on certain features or attributes.
- The similarity or dissimilarity between data points is often measured using distance metrics.
- The algorithm's objective is to minimize the intra-cluster distance (distance between data points within the same cluster) while maximizing the inter-cluster distance (distance between data points from different clusters)



Clustering key concepts





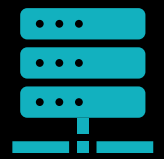
1. **Distance Metric:** A measure of how similar or dissimilar two data points are. Common distance metrics include Euclidean distance, Manhattan distance, and cosine similarity.

2. **Centroids or Medoids:** Clusters are often defined around central points, which can be represented by centroids (average values of the data points) or medoids (actual data points).

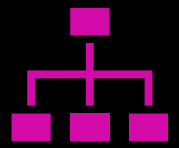
3. **Cluster Assignment:** Data points are assigned to clusters based on their proximity to cluster centers. Each data point belongs to exactly one cluster.

4. **Number of Clusters:** The number of clusters is often specified beforehand, but in some cases, the algorithm might determine the optimal number of clusters based on the data or user input.

Clustering Algorithms



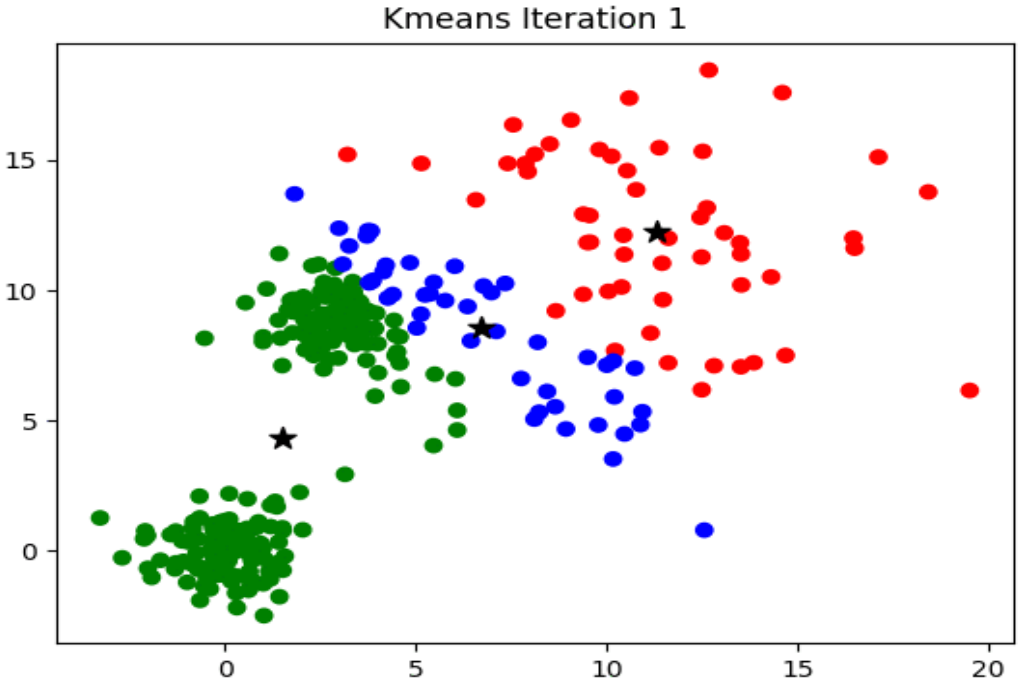
K-Means Clustering: Partitioning data into k clusters by iteratively updating cluster centroids. ✅

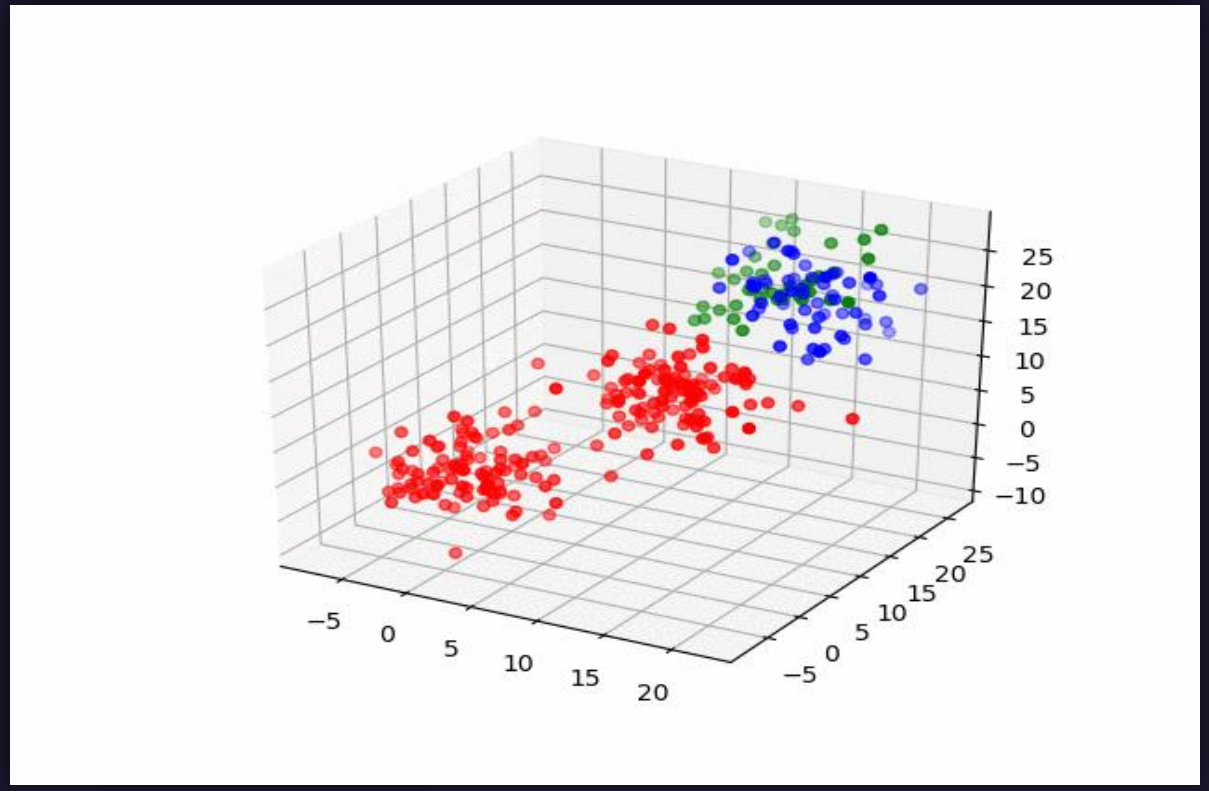


Hierarchical Clustering: Creating a hierarchy (similar to trees) of clusters by iteratively merging or splitting clusters.

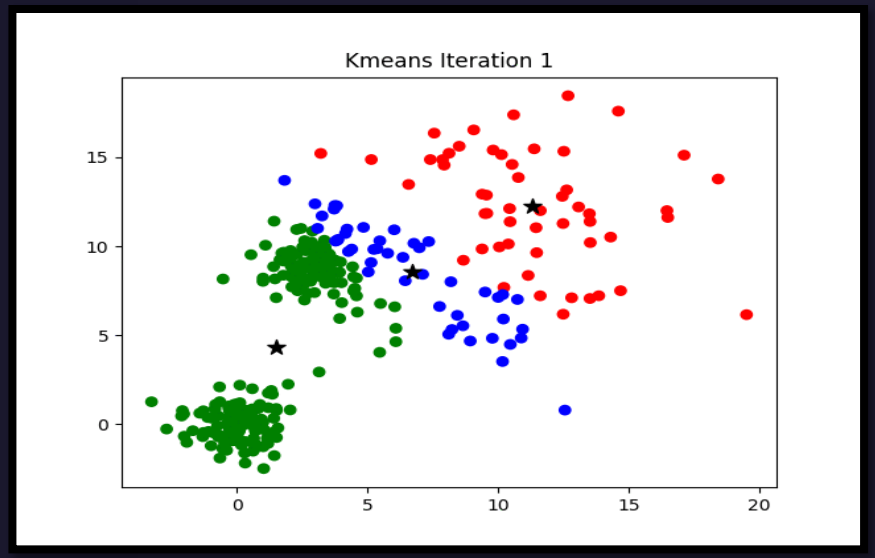
DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Identifying clusters based on data density and noise.

K-means Algorithm





K-means Algorithm



1. Initialization
    - Choose the number of clusters k
    - Initialize *k* cluster centers randomly.
2. Assignment
    - For each data point, calculate the distance to each of the k clusters centers.
    - Assign the data point to the cluster whose center is close to it.
3. Update
    - Recalculate the cluster centers by taking the mean of all data points assigned to each cluster. This moves the cluster center to the "center of mass" of the cluster's data points.
4. Convergence check
    - Check if the cluster centers have changed significantly compared to the previous iteration. If not, the algorithm has converged and the process stops. Otherwise, go back to the Assignment Step

K-means limitations

The number of clustering k need to be specified.

K-means assumes that clusters are spherical and equally sized, which may not hold for all datasets.

Outliers can heavily influence cluster centers and lead to suboptimal results.

K-means may struggle with non-linear or complex cluster shapes.

K-means How to choose K?

Domain Knowledge:

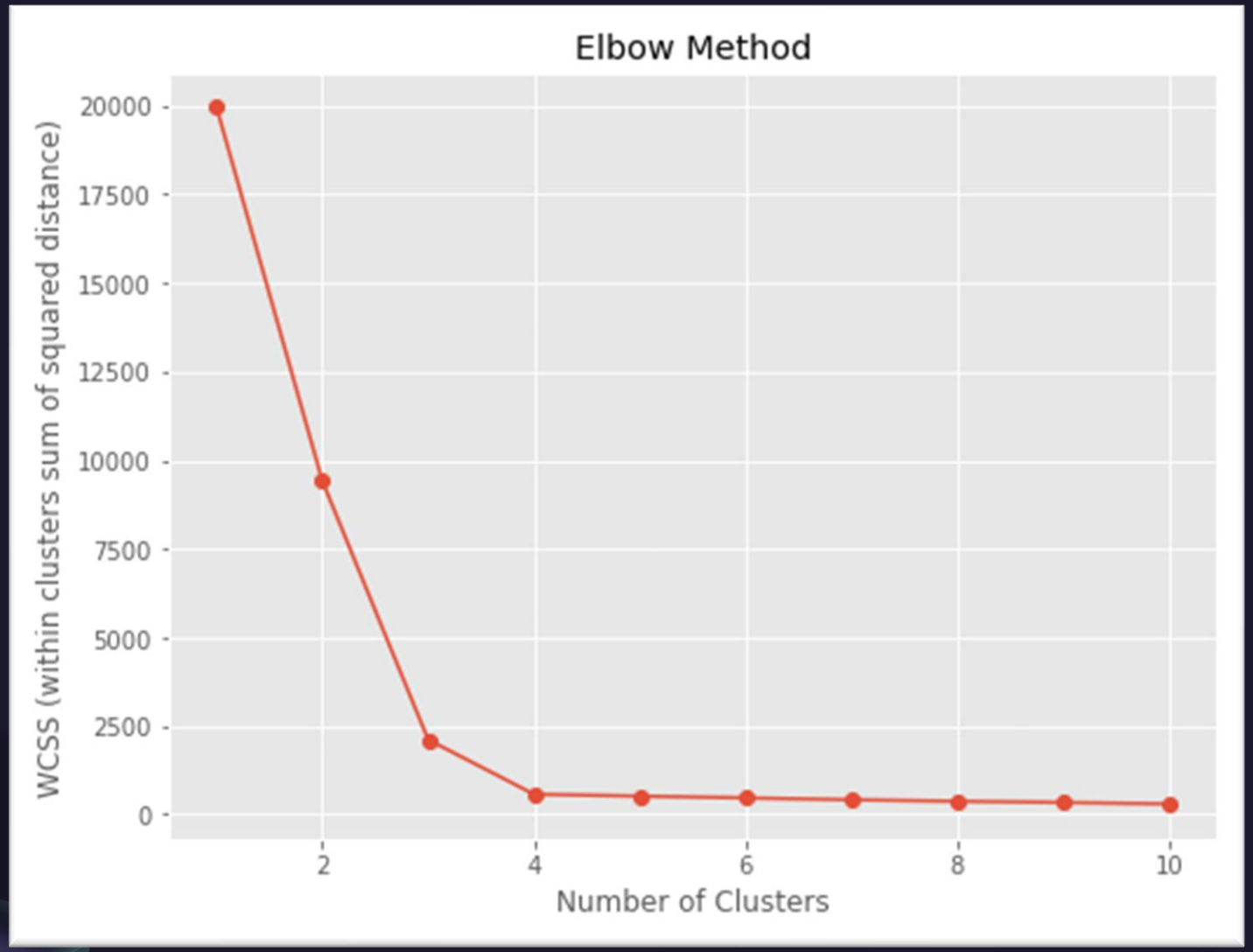
- If you have prior knowledge about the problem domain, you might have insights into the expected number of natural clusters.

Visualization

Elbow method :

- The idea is that to plot average distance in each cluster for different values of k
- When adding more k clusters and no change is seeing that mean it’s useless to add more centroid that won’t reduce the variance between clusters

Elbow method



Elbow method

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

# Generate sample data
data, _ = make_blobs(n_samples=300, centers=4, random_state=42)

# Calculate WCSS for different values of k
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)

# Plot the elbow curve
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (within clusters sum of squared distance)')
plt.show()
```

Dimensionality reduction curse of dimensionality

- **Sparsity of Data** ❌

  _ more dimensions and less points.

- **Increased Computational Complexity** ❌
- **Distance and Similarity** ❌
- **Overfitting** ❌
- **Data Visualization** ❌
- Feature Selection and Engineering ❌

Dimensionality reduction
Principal Component Analysis

PCA aims to find a **set of orthogonal axes** (principal components) in the original feature space that **capture the maximum variance in the data**. These axes are ordered by the amount of variance they explain. The first principal component captures the most variance, the second captures the second most, and so on.

Dimensionality reduction
Principal Component Analysis

```python
import numpy as np
# Example data matrix
X = np.array([[1, 2, 3, 5, 3],
              [4, 5, 6, 6, 7],
              [7, 8, 9, 10, 11]])

# Step 1: Standardize the data
X_std = (X - np.mean(X, axis=0)) / np.std(X, axis=0)
# Step 2: Calculate the covariance matrix
C = np.cov(X_std, rowvar=False)
# Step 3: Eigenvalue decomposition
eigenvalues, eigenvectors = np.linalg.eig(C)
# Step 4: Choosing principal components
k = 3  # Number of principal components to keep
top_eigenvectors = eigenvectors[:, :k]
# Step 5: Project data onto the new subspace
X_pca = X_std @ top_eigenvectors
# Resulting reduced data
print(X_pca, X_pca.shape)
```

Dimensionality reduction
Principal Component Analysis

```python
import numpy as np
from sklearn.decomposition import PCA

# Generate random data
np.random.seed(0)
X = np.random.rand(10, 5)  # 10 samples, 5 features

# Instantiate PCA and transform data
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Access principal components and explained variance ratio
principal_components = pca.components_

print("Original Data Shape:", X.shape)
print("Reduced Data Shape:", X_pca.shape)
print("Principal Components:", principal_components)
```

```
Original Data Shape: (10, 5)
Reduced Data Shape: (10, 2)
...
```

Dimensionality reduction
Independent Component Analysis

ICA is based on the idea of **separating a set of mixed signals** into their original, statistically **independent sources**. It assumes that the observed signals are linear mixtures of these sources, and the goal is to find a transformation that "unmixes" the signals.

Dimensionality reduction
Independent Component Analysis

```python
import numpy as np
from sklearn.decomposition import FastICA

# Instantiate ICA and unmix the sources
ica = FastICA(n_components=3)
sources = ica.fit_transform(mixed_data)

# Print the first few rows of the unmixed sources
print("Unmixed Sources:")
print(sources[:5, :])
```

See 👀

- https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/
- http://alekseynp.com/viz/k-means.html
- https://scikitlearn.org/stable/auto_examples/cluster/plot_cluster_comparison.html
- https://www.naftaliharris.com/blog/visualizing-k-means-clustering/ ✨
- https://clustering-visualizer.web.app/ ✨

References

- https://en.wikipedia.org/wiki/Covariance_matrix
- https://www.cs.helsinki.fi/u/ahyvarin/papers/NN00new.pdf
- https://youtu.be/FgakZw6K1QQ
- https://towardsdatascience.com/independent-component-analysis-ica-a3eba0ccec35
- https://eeglab.org/tutorials/06_RejectArtifacts/RunICA.html