Clustering Algorithms Continued

Minor in AI, IIT Ropar

14th May, 2025

1 Customer Segmentation

Background

An online retail chain wants to segment its customers to target marketing campaigns more effectively. The dataset contains customer purchase histories, frequency of visits, and demographic information.

Figure 1: Image Source

Hierarchical Clustering

Scenario:
The marketing team wants to understand how customers naturally group together without predefining the number of segments.

Approach 1: Agglomerative Clustering

- Each customer starts as their own cluster.
- Clusters are merged based on similarity in purchase behavior and demographics.

  Approach 2: Divisive Clustering

- All customers are in a single cluster.
- Clusters are broken down based on similarity in purchase behavior and demographics.

  Visualization:

- A **dendrogram** can be plotted to visualize hierarchical clustering.
- By cutting (horizontally) the dendrogram at a certain height, one can indentify different customer segments. For eg. cutting at a line where 3 segments are found such as:
  1. High spenders, frequent visitors
  2. Occasional shoppers
  3. Discount seekers

  Business Impact:

- Tailored promotions can be sent to each segment (after proper customer segmentation), increasing campaign effectiveness.

Linkage Methods

Scenario:
The team tests different linkage methods to see which best separates customer types.

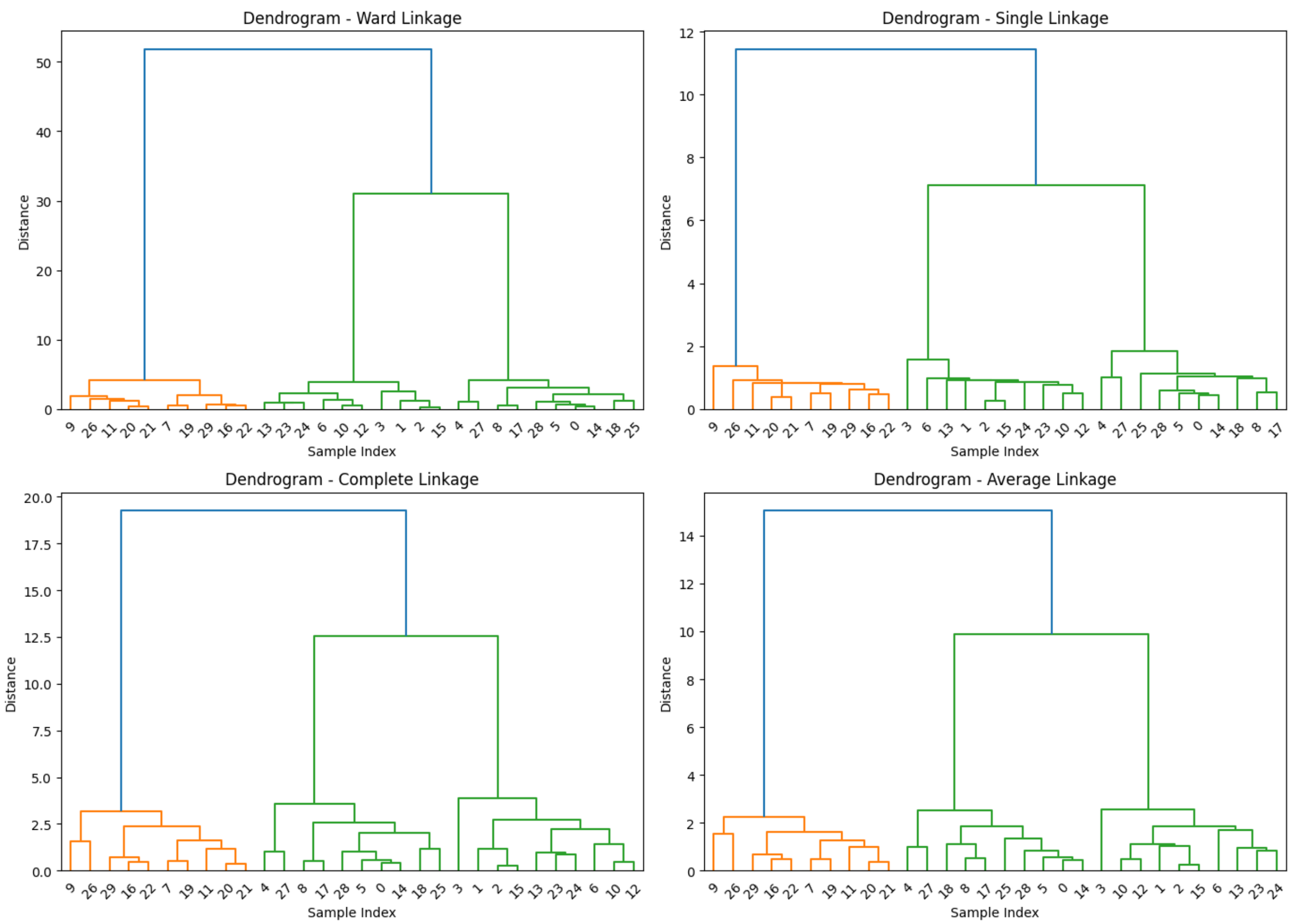
- **Single Linkage:** Groups customers who share at least one similar purchase, but results in elongated, less meaningful clusters.
- **Complete Linkage:** Groups only those who are very similar, leading to tighter, more actionable segments.
- **Average Linkage:** Groups based on balanced approach, leading to moderate segments.
- **Ward Linkage:** Minimizes variance within clusters, producing balanced, interpretable segments.

Feature	Single Linkage	Complete Linkage	Average Linkage	Ward Linkage
Distance Metric	Minimum distance between points in clusters	Maximum distance between points in clusters	Average of all pairwise distances	Increase in total within-cluster variance
Tends to Form	Long chains ("chaining" effect)	Compact, spherical clusters	Balanced between chaining and compactness	Compact, equally sized clusters
Sensitive to Noise?	Yes – outliers easily pull in points	No – more robust to noise	Somewhat	No – least sensitive
Cluster Shape	Arbitrary, elongated	Spherical or compact	Balanced shape flexibility	Spherical, equal-sized clusters
Computational Cost	Low	Medium	Medium	Higher
Best Use Case	Finding elongated or nested clusters	When compact and well-separated clusters exist	When cluster sizes vary moderately	When you expect compact, equally sized clusters
Worst Case	Chains form even if clusters are distinct	May split true clusters	Can average out real separation	Doesn't perform well on non-spherical clusters
Agglomeration Style	Greedy (connect nearest neighbors)	Greedy (connect farthest points)	Mid-range strategy	Variance-based, statistically driven

2 Dendrograms: Visualizing Cluster Merges

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from scipy.cluster.hierarchy import linkage, dendrogram

# Step 1: Generate synthetic dataset
X, _ = make_blobs(n_samples=30, centers=3, random_state=42)

# Step 2: List of linkage methods
methods = ['ward', 'single', 'complete', 'average']

# Step 3: Set up a 2x2 plot grid
fig, axs = plt.subplots(2, 2, figsize=(14, 10))
axs = axs.flatten()

# Step 4: Loop through each method and plot the dendrogram
for i, method in enumerate(methods):
    Z = linkage(X, method=method)
    dendrogram(Z, ax=axs[i])
    axs[i].set_title(f'Dendrogram - {method.capitalize()} Linkage')
    axs[i].set_xlabel('Sample Index')
    axs[i].set_ylabel('Distance')
```

```python
plt.tight_layout()
plt.show()
```



Dendrograms for different linkage methods applied during clustering on a synthetic data set.

3 DBSCAN

Core Concepts of DBSCAN

- `eps` (**epsilon**): Defines the radius of a point's neighborhood. It controls how close points must be to each other to be considered part of the same cluster. The choice of `eps` is critical:
  - A small `eps` may result in many points being labeled as outliers or forming fragmented, tiny clusters.
  - A large `eps` may cause distinct clusters to merge, reducing the meaningfulness of the clustering.
- `min_samples`: The minimum number of points required within the `eps` neighborhood for a point to be considered a **core point**. Core points are the foundation of clusters in

DBSCAN.

- **Core point**: A point with at least `min_samples` points (including itself) within its `eps` neighborhood.
- **Border point**: A point that is within the `eps` neighborhood of a core point but does not itself have enough neighbors to be a core point.
- **Noise point**: Any point that is not a core point or a border point is labeled as noise (assigned the label `-1`).
- The values of `eps` and `min_samples` directly affect the number, size, and shape of clusters detected. Choosing the right `eps` is crucial and is often done through visual inspection or using a k-distance plot.
- DBSCAN is powerful for detecting clusters of arbitrary shape and is robust to noise and outliers.

DBSCAN on 2-Moons Dataset

Approach:

- The **two-moons** synthetic dataset is generated to simulate two interleaved, non-convex clusters, a classic challenge for traditional clustering algorithms.
- **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) is applied with `eps=0.2` and `min_samples=5`, enabling the detection of clusters based on density and automatic identification of noise points.
- The dataset contains 200 samples with slight noise (`noise=0.05`) to mimic real-world imperfections.

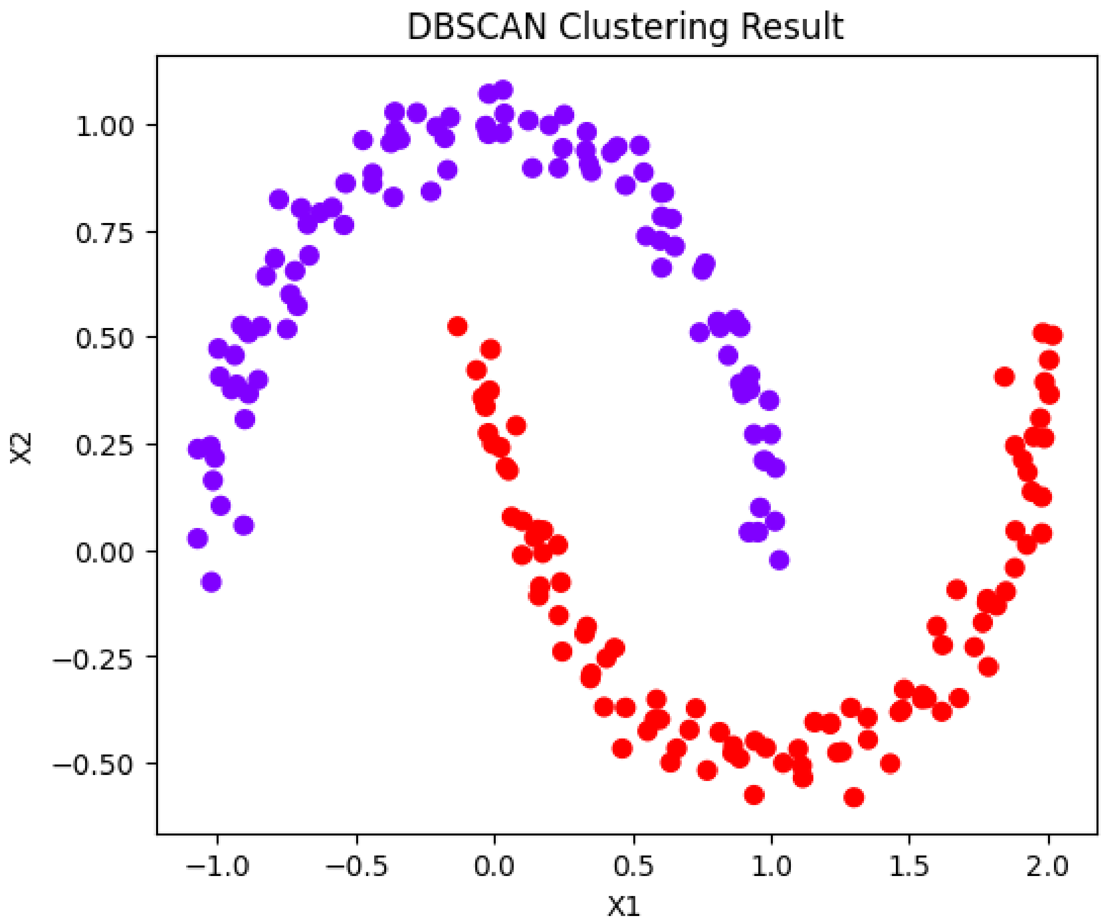
  Python Code:

```
# Measuring Accuracy
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score, davies_bouldin_score

# Step 1: Generate two-moons dataset
X, _ = make_moons(n_samples=200, noise=0.05, random_state=0)

# Step 2: Apply DBSCAN
dbscan = DBSCAN(eps=0.2, min_samples=5)
labels = dbscan.fit_predict(X)
```

```
# Step 3: Plot clustering result
plt.figure(figsize=(6, 5))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='rainbow', s=40)
plt.title("DBSCAN Clustering Result")
plt.xlabel("X1")
plt.ylabel("X2")
plt.show()

# Step 4: Filter out noise for evaluation (-1 are noise points)
mask = labels != -1
X_core = X[mask]
labels_core = labels[mask]

# Step 5: Evaluation
if len(set(labels_core)) > 1:
    sil_score = silhouette_score(X_core, labels_core)
    db_score = davies_bouldin_score(X_core, labels_core)
    print(f"Silhouette Score: {sil_score:.2f}")
    print(f"Davies-Bouldin Index: {db_score:.2f}")
else:
    print("Not enough clusters (or too many noise points) for evaluation.")
```

Visualization:

- The resulting clusters are visualized in a scatter plot, where each color represents a different cluster assigned by DBSCAN.
- Noise points (if any) are colored differently or omitted, clearly distinguishing outliers from core cluster members.
- The plot below demonstrates DBSCAN's ability to correctly separate the two moon-shaped clusters, which are not linearly separable.



Evaluation Metrics:

- Before evaluation, noise points (labeled as `-1`) are filtered out to focus on the quality of the actual clusters.
- **Silhouette Score:** Measures how similar each point is to its own cluster compared to other clusters. A higher score indicates well-defined clusters.
- **Davies-Bouldin Index:** Evaluates the average similarity between clusters, where a lower value indicates better clustering.
- In this case, both metrics confirm that DBSCAN has effectively identified the two non-convex clusters, outperforming algorithms like K-means on this dataset.

Outcome:

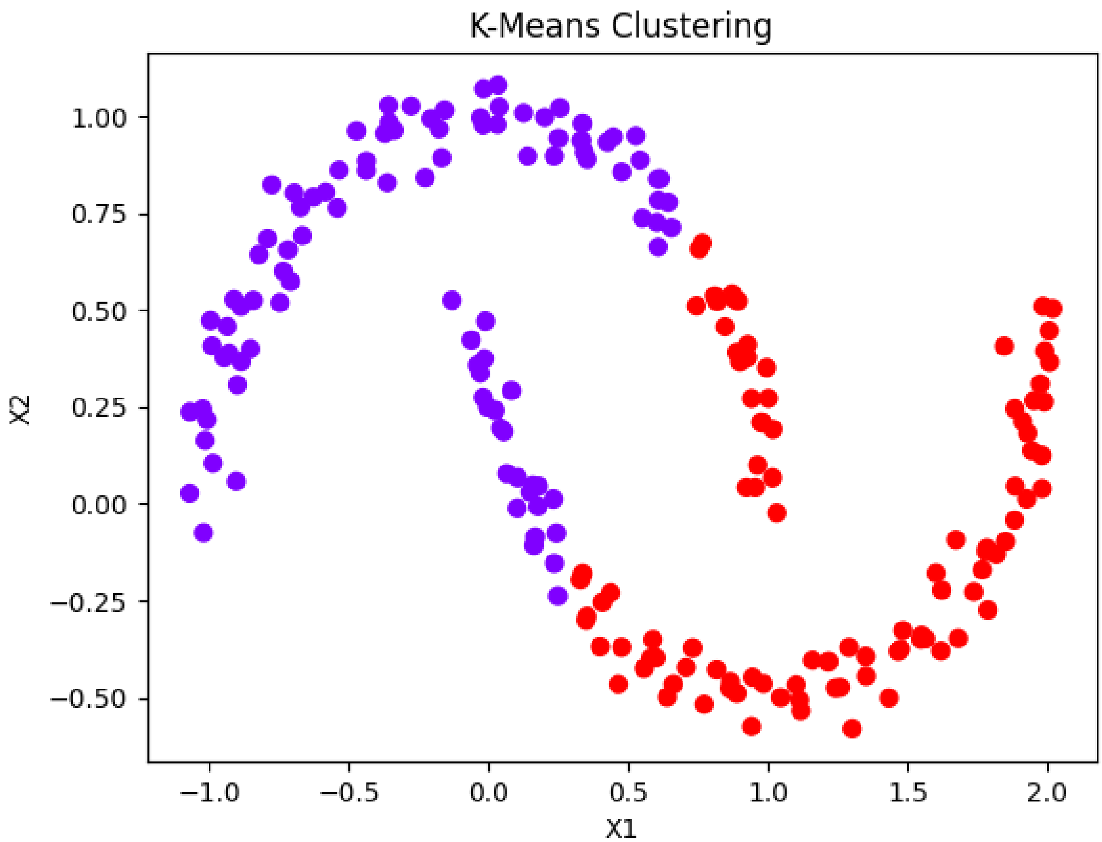
- DBSCAN successfully separates the two interleaved moon-shaped clusters without requiring the number of clusters as input.
- The approach is robust to noise and can handle irregular cluster shapes, making it suitable for complex real-world data distributions.
- Quantitative metrics (Silhouette Score and Davies-Bouldin Index) provide objective validation of clustering quality.

K-Means Clustering v/s DBSCAN on 2-Moons Dataset

As we can see in visualization of below Python code, K-Means won't be able to segment the datapoints properly.

```
# Applying DBSCAN on 2-moons data set
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
import numpy as np
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans


# Understand the dataset
X, _ = make_moons(n_samples=200, noise=0.05, random_state=0)
plt.scatter(X[:, 0], X[:, 1])
plt.title("Two Moons Dataset")
plt.xlabel("X1")
plt.ylabel("X2")
plt.show()


# What would k-means do to this?
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans_labels = kmeans.fit_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='rainbow', s=40)
plt.title("K-Means Clustering")
plt.xlabel("X1")
plt.ylabel("X2")
plt.show()

# DBSCAN with chosen parameters
db = DBSCAN(eps=0.2, min_samples = 5)
db.fit(X)

# Get labels (-1 means noise)
labels = db.labels_

# Plotting
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='rainbow', s=40)
plt.title("DBSCAN Clustering")
plt.xlabel("X1")
plt.ylabel("X2")
plt.show()
```

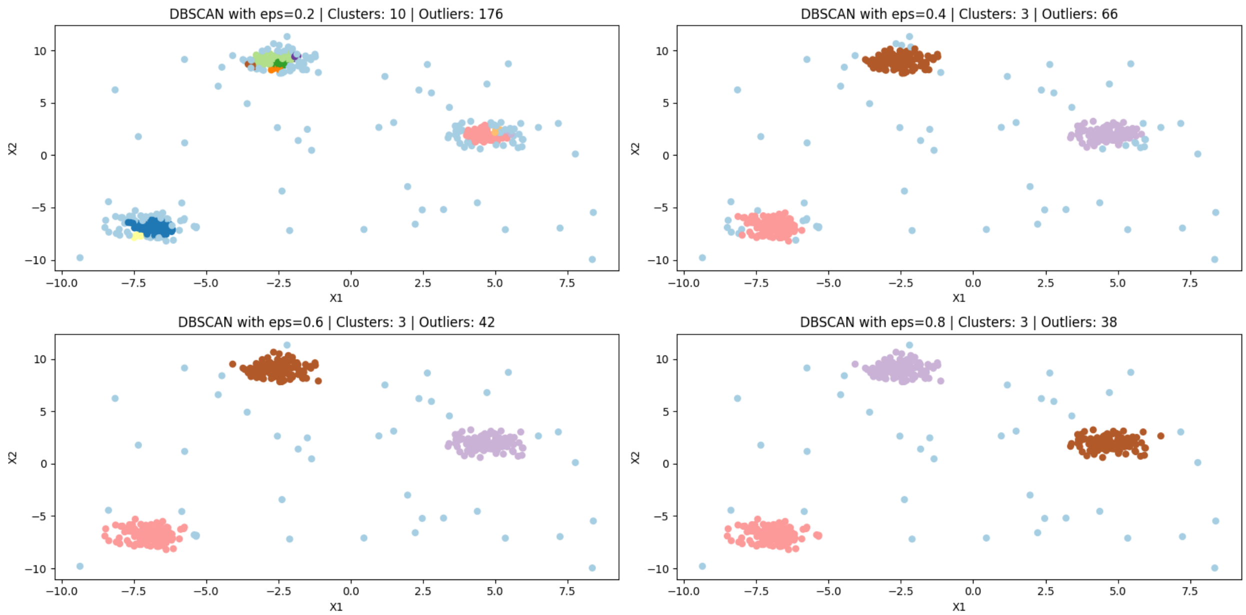
```
# Filter out noise points
core_samples_mask = labels != -1
if np.sum(core_samples_mask) > 1:
    score = silhouette_score(X[core_samples_mask], labels[core_samples_mask])
    print(f"Silhouette Score: {score:.2f}")
else:
    print("Not enough core samples for silhouette score.")
```



Effect of varying parameters of DBSCAN

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score, davies_bouldin_score

# Step 1: Generate synthetic clustered data with noise
X, _ = make_blobs(n_samples=400, centers=3, cluster_std=0.6, random_state=42)
noise = np.random.uniform(low=-10, high=10, size=(40, 2))  # Add some outliers
```

```
X = np.vstack([X, noise])  # Combine clusters and outliers

# Step 2: Try DBSCAN with different eps values
# In DBSCAN, eps (epsilon) defines the radius of a point's neighborhood.
# It controls how close points must be to be considered part of the same cluster.
# A point is labeled a core point if it has enough neighbors (min_samples)
# within this eps radius.
# Small eps values can lead to many outliers or fragmented clusters, while
# large values may merge distinct clusters.
# It directly affects the number and shape of clusters detected.
# Choosing the right eps is crucial and often done through
# visual inspection or a k-distance plot.
# DBSCAN is powerful for detecting clusters of arbitrary shape and noise.
eps_values = [0.2, 0.4, 0.6, 0.8]

# Step 3: Plot the results
plt.figure(figsize=(16, 8))
for i, eps in enumerate(eps_values, 1):
    dbscan = DBSCAN(eps=eps, min_samples=5)
    labels = dbscan.fit_predict(X)

    plt.subplot(2, 2, i)
    plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='Paired', s=30)
    plt.title(f'''DBSCAN with eps={eps} | Clusters: {len(set(labels)) -
    (1 if -1 in labels else 0)} | Outliers: {np.sum(labels==-1)}''')
    plt.xlabel("X1")
    plt.ylabel("X2")

plt.tight_layout()
plt.show()
```



4 Key Takeaways from Case Studies

- **Clustering helps uncover hidden patterns** in diverse domains: marketing, public safety, content organization.
- **Choice of algorithm and parameters** depends on data shape, business goals, and interpretability needs.
- **Visualization (dendrograms, maps, scatter plots)** is crucial for understanding and communicating results.
- **Evaluation metrics** guide model selection but must be complemented by domain knowledge.

Check out this google sheet for extra info.

5 Appendix

Davies-Bouldin Index

$$DB = \frac{1}{n}\sum_{i=1}^{n} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \tag{1}$$

The Davies-Bouldin Index evaluates clustering quality where:

- $\sigma_i and \sigma_j$ represent the average distance of points to their cluster centers
- $d(c_i, c_j)$ is the distance between cluster centers
- Lower values indicate better clustering with more distinct separation