

▼ **AML Lab Assignment-5:**

Name: Gurvinder Kaur Matharu  
PRN: 1032230432  
Roll No.: PA14

**AIM:**

Implementation and comparison of various Clustering Techniques: Spectral and DBSCAN

**THEORY:**

Clustering is the process of grouping similar data points together in a dataset, aiming to identify inherent patterns or structures without prior knowledge of group membership, often used in unsupervised machine learning.

Spectral clustering is a technique that uses the eigenvalues and eigenvectors of a similarity matrix derived from data points to partition them into clusters. It embeds the data into a lower-dimensional space where clusters are more separable, allowing for effective clustering even in complex, non-linear datasets. By utilizing spectral properties, it identifies relationships between data points based on their spectral decomposition, often outperforming traditional clustering methods in certain scenarios.


DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that identifies clusters in data based on density. It groups together data points that are closely packed, defining clusters as areas with high density separated by regions of lower density. It doesn't require specifying the number of clusters beforehand and can detect outliers as noise points, making it effective for various shapes and sizes of clusters in datasets.

▼ **CODE EXECUTION & OUTPUT:**



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN, SpectralClustering
from sklearn import metrics
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

```
data = pd.read_csv('/content/Mall_Customers.csv')
```

```
data.head()
```



	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40



```
data.shape
```

(200, 5)

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   CustomerID            200 non-null   int64
1   Gender                200 non-null   object
2   Age                  200 non-null   int64
3   Annual Income (k$)    200 non-null   int64
4   Spending Score (1-100) 200 non-null   int64
```

```
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
data.isnull().sum()
```

```
CustomerID      0
Gender          0
Age            0
Annual Income (k$)  0
Spending Score (1-100)  0
dtype: int64
```

```
X=data.iloc[:,[2,4]].values
```

## DBSCAN Clustering

```
dbscan = DBSCAN(eps=5, min_samples=6)
dbscan_model = dbscan.fit(X)
dbscan_labels = dbscan_model.labels_
```

```
dbscan_sample_cores = np.zeros_like(dbscan_labels, dtype=bool)
dbscan_sample_cores[dbscan.core_sample_indices_] = True
# Calculating the number of clusters for DBSCAN
dbscan_n_clusters = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
print("DBSCAN silhouette score: ", metrics.silhouette_score(X, dbscan_labels))
```

```
DBSCAN silhouette score:  0.19686977554902063
```

## Spectral Clustering

```
spectral_model = SpectralClustering(n_clusters=dbscan_n_clusters, affinity='nearest_neighbors')
spectral_labels = spectral_model.fit_predict(X)
print("Spectral Clustering silhouette score: ", metrics.silhouette_score(X, spectral_labels))
```

```
Spectral Clustering silhouette score:  0.42389696192962406
```

```
# Plotting the clusters
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# DBSCAN plot
unique_labels = set(dbscan_labels)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]
    class_member_mask = (dbscan_labels == k)
    xy = X[class_member_mask & dbscan_sample_cores]
    ax[0].plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col), markeredgecolor='k', markersize=14)
    xy = X[class_member_mask & ~dbscan_sample_cores]
    ax[0].plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col), markeredgecolor='k', markersize=6)
ax[0].set_title('DBSCAN: Estimated number of clusters: %d' % dbscan_n_clusters)

# Spectral Clustering plot
unique_labels = set(spectral_labels)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]
    class_member_mask = (spectral_labels == k)
    xy = X[class_member_mask]
    ax[1].plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col), markeredgecolor='k', markersize=14)
ax[1].set_title('Spectral Clustering')

plt.show()
```

