Machine Learning for Data Science

M. H. Rahman

Associate Professor
Department of Statistics and Data Science
Jahangirnagar University
Bangladesh
E-mail: habib.drj@juniv.edu

Twenty-Fifty

Course Code: PM-ASDS 22
Course Title: Machine Learning for Data Science
Department of Statistics
Jahangirnagar University

Name of the Topics of this Course

- Overview of Machine Learning for Data Science
- Statistics and Machine Learning
- Machine Learning Techniques
- Neural Network
- Classification
- Clustering
- Naive Bayes

Book References

Texts

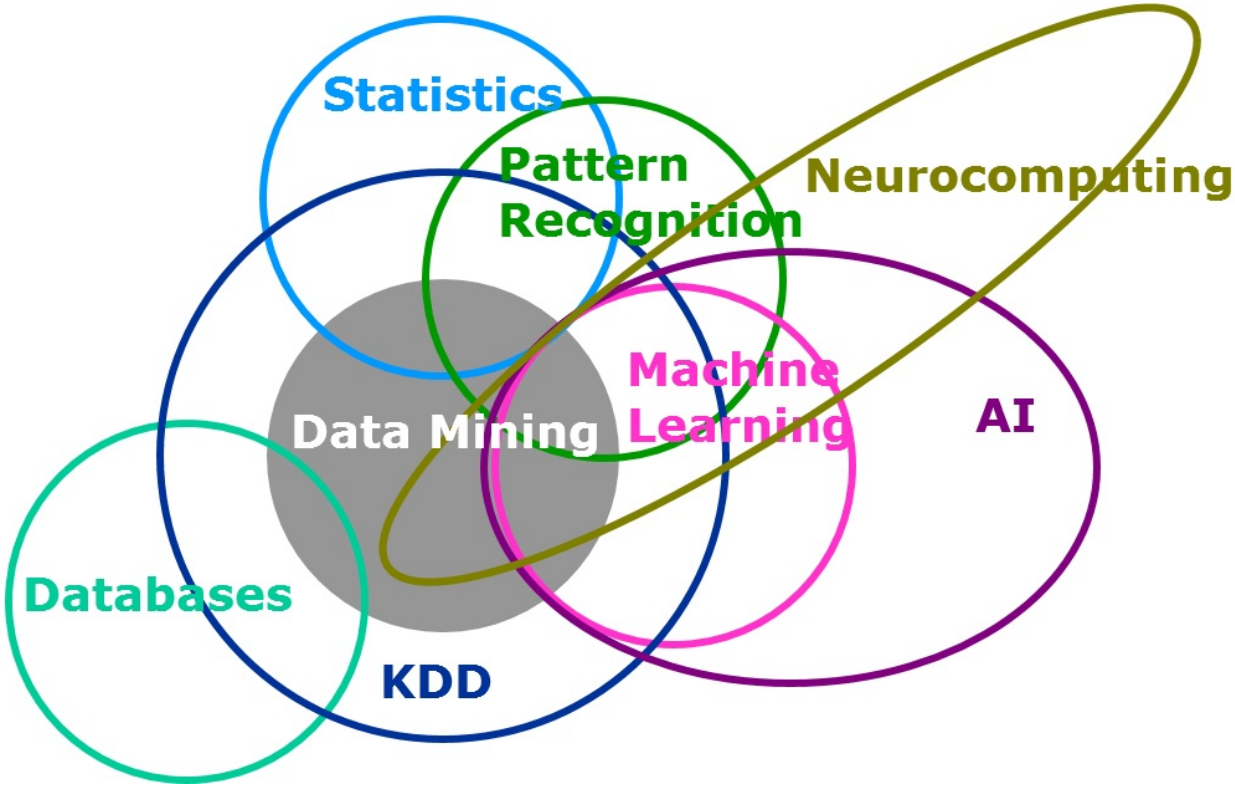
1. Daniel D. Gutierrez, (2015): Machine Learning and Data Science: An Introduction to Statistical Learning Methods with R, 1st Edition, Technics Publications.
2. Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., (2009): The elements of statistical learning: data mining, inference, and prediction. New York: Springer.
3. John Paul Mueller and Luca Massaron (2016): Machine Learning For Dummies, 1st Edition, For Dummies.

References

1. Han J., Kamber M., Pei J (2011): Data Mining: Concepts and Techniques, 3rd edition, Elsevier.

Machine Learning for Data Science

Machine Learning for Data Science



The figure is collected from the internet.

Spectral Clustering

Spectral Clustering

Graph Theory

In mathematics, graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices (also called nodes or points) which are connected by edges (also called links or lines). A distinction is made between undirected graphs, where edges link two vertices symmetrically, and directed graphs, where edges link two vertices asymmetrically. Graphs are one of the principal objects of study in discrete mathematics (Source: Wikipedia).

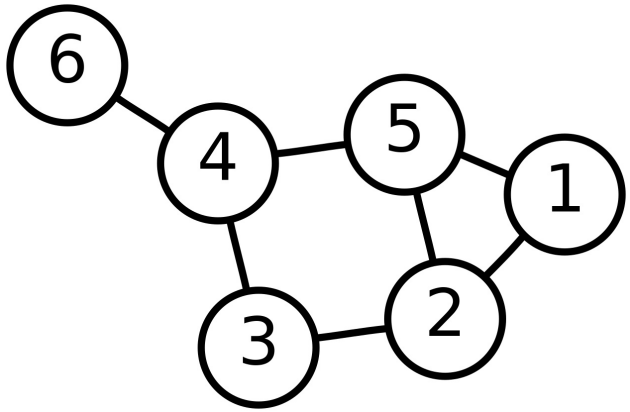


Figure: A drawing of a graph (Sorce: Wikipedia).

Spectral Clustering

Spectral Graph Theory

In mathematics, spectral graph theory is the study of the properties of a graph in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of matrices associated with the graph, such as its adjacency matrix or Laplacian matrix.

The adjacency matrix of a simple undirected graph is a real symmetric matrix and is therefore orthogonally diagonalizable; its eigenvalues are real algebraic integers.

While the adjacency matrix depends on the vertex labeling, its spectrum is a graph invariant, although not a complete one.

Spectral graph theory is also concerned with graph parameters that are defined via multiplicities of eigenvalues of matrices associated to the graph

Spectral Clustering

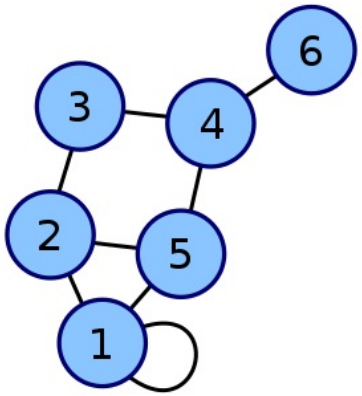
Laplacian Matrix or Adjacency matrix

In the mathematical field of graph theory, the Laplacian matrix, also called the graph Laplacian, admittance matrix, Kirchhoff matrix or discrete Laplacian, is a matrix representation of a graph.

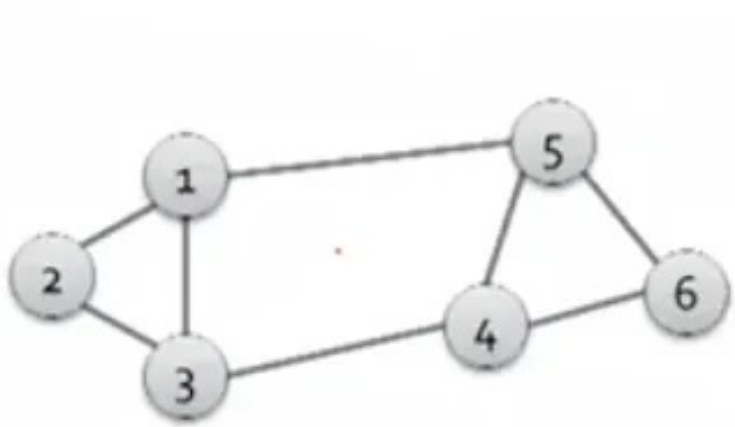
In graph theory and computer science, an adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.

In the special case of a finite simple graph, the adjacency matrix is a (0,1)-matrix with zeros on its diagonal. If the graph is undirected (i.e. all of its edges are bidirectional), the adjacency matrix is symmetric. The relationship between a graph and the eigenvalues and eigenvectors of its adjacency matrix is studied in spectral graph theory.

Spectral Clustering

Labeled graph	Adjacency matrix
	$\begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Spectral Clustering



	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

Figure: Adjacency Matrix.

Spectral Clustering

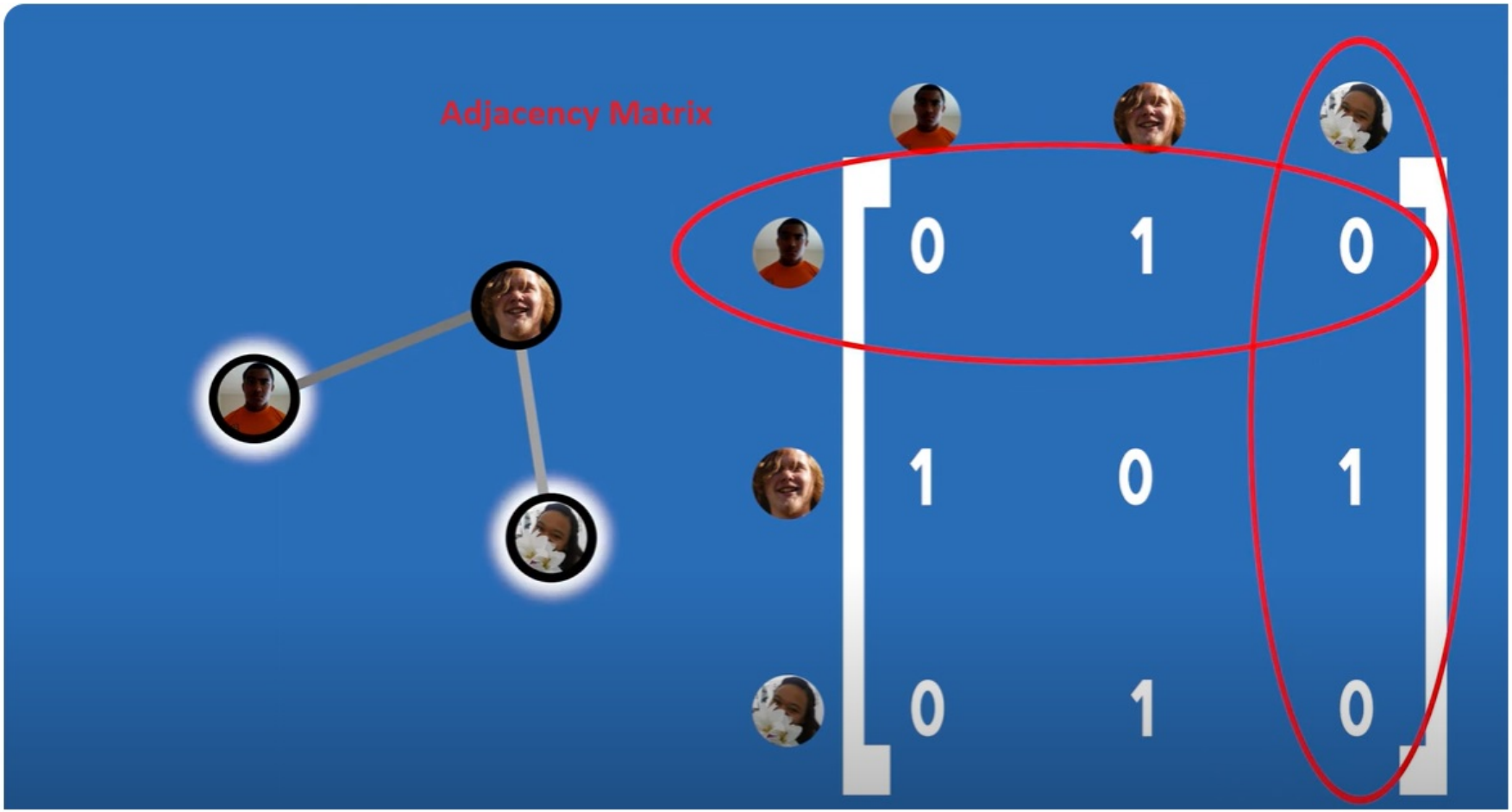


Figure: Adjacency Matrix.

Spectral Clustering

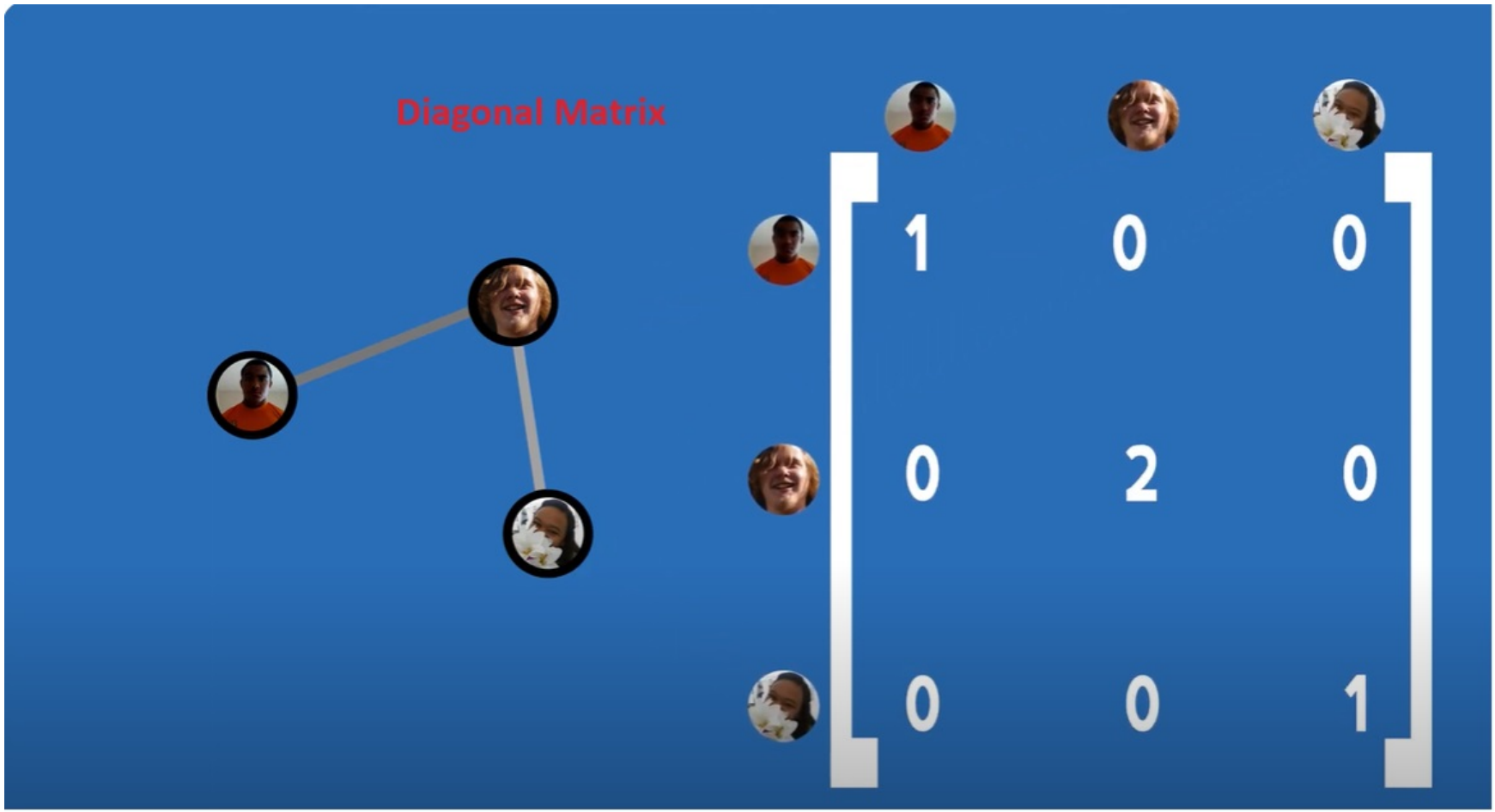


Figure: Diagonal Matrix.

Spectral Clustering

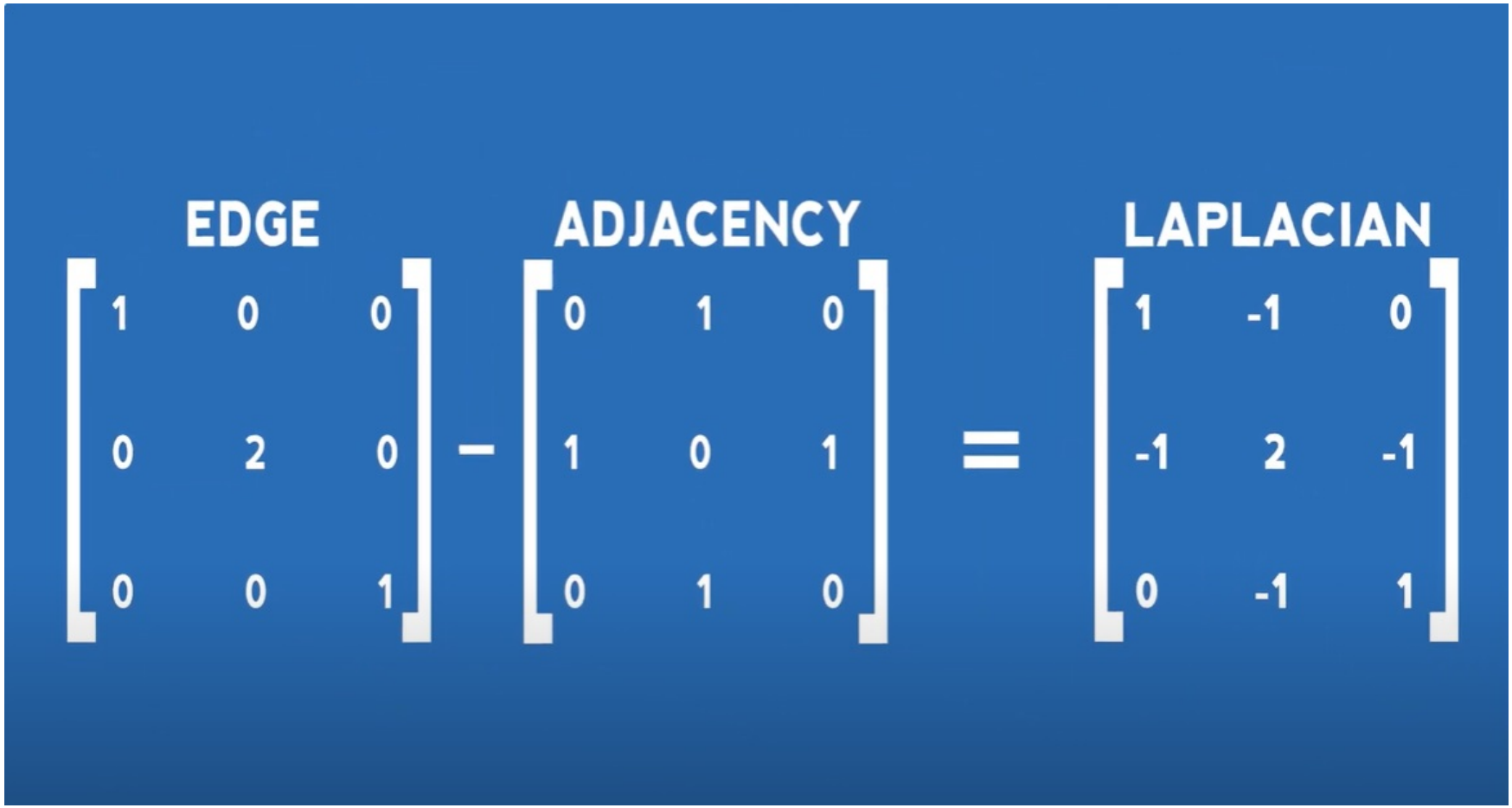


Figure: Laplacian Matrix.

Spectral Clustering

Laplacian matrix

Laplacian matrix defined as

$L = D - A$

where, D is the diagonal matrix, and A is the adjacency matrix.

Laplacian matrix normalization

Spectral Clustering

The steps involved in spectral clustering in more detail:

1. Construct Affinity Matrix: Given a dataset with n data points, construct an $n \times n$ affinity matrix W, where W_{ij} represents the similarity or affinity between data points i and j. Common similarity measures include Gaussian kernel, k-nearest neighbors, or cosine similarity.
2. Degree Matrix: Compute the degree matrix D, a diagonal matrix where D_{ii} is the sum of the similarities between data point i and all other data points: $D_{ii} = \sum_j W_{ij}$.
3. Normalized Laplacian Matrix: Compute the normalized Laplacian matrix L_{norm} using the formula $L_{\text{norm}} = I - D^{-1/2} W D^{-1/2}$, where I is the identity matrix. This step ensures that the Laplacian matrix is symmetric and positive semi-definite.

Spectral Clustering

4. Eigenvalue Decomposition: Perform eigenvalue decomposition on the normalized Laplacian matrix L_{norm} to obtain its eigenvectors and corresponding eigenvalues. Let $\lambda_1, \lambda_2, ..., \lambda_n$ be the eigenvalues and $v_1, v_2, ..., v_n$ be the corresponding eigenvectors.
5. Select Number of Clusters: Choose the number of clusters k either based on domain knowledge or using techniques like the eigengap heuristic, silhouette score, or gap statistic.
6. Low-dimensional Representation: Select the k eigenvectors corresponding to the k smallest eigenvalues and form a matrix V with these eigenvectors as columns.

Spectral Clustering

7. Cluster Assignment: Apply traditional clustering algorithms (e.g., K-means) to the rows of the matrix V to assign data points to k clusters. Alternatively, spectral clustering can also use other methods for clustering such as spectral embedding or normalized cuts.
8. Output: Return the cluster assignments for each data point.

These steps outline the general procedure for spectral clustering. Depending on implementation and variations, additional steps or optimizations may be included. Additionally, spectral clustering can be adapted to various scenarios and data types, making it a versatile technique in machine learning and data analysis.

Spectral Clustering

Let's go through an example of spectral clustering step by step:
Suppose we have a dataset with five data points in 2-dimensional space:

$$A(7.65, 7.97), B(7.65, 8.56), C(7.82, 8.77), D(8.79, 8.32), E(8.72, 7.48)$$

We'll use a Gaussian similarity measure to construct the similarity matrix. The similarity between two points i and j is calculated as:

$$K(x_i, x_j) = \text{similarity}(i, j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

$$\|x_i - x_j\|^2 = \sum_{k=1}^{p}(x_{ik} - x_{jk})^2$$

where x_i and x_j are the feature vectors of points i and j, and σ is a parameter controlling the width of the Gaussian.

Spectral Clustering

The Euclidean distance matrix:

	A	B	C	D	E
A	0.000	0.590	0.818	1.193	1.177
B	0.590	0.000	0.270	1.165	1.520
C	0.818	0.270	0.000	1.069	1.573
D	1.193	1.165	1.069	0.000	0.843
E	1.177	1.520	1.573	0.843	0.000

Let's assume $\sigma = 1$, then the Gaussian similarity matrix, W becomes:

	A	B	C	D	E
A	1.000	0.840	0.716	0.491	0.500
B	0.840	1.000	0.964	0.507	0.315
C	0.716	0.964	1.000	0.565	0.290
D	0.491	0.507	0.565	1.000	0.701
E	0.500	0.315	0.290	0.701	1.000

Spectral Clustering

1. Graph Construction:
Based on this similarity matrix, we construct a weighted graph where each data point is a node and edges represent the similarity between data points.

2. Laplacian Matrix:
Next, we compute the Laplacian matrix. We have several choices for the Laplacian matrix, such as unnormalized, normalized, or symmetric normalized Laplacian. Let's use the normalized Laplacian matrix for this example.

The normalized Laplacian matrix L is defined as $L = I - D^{-1/2}WD^{-1/2}$, where D is the degree matrix (a diagonal matrix with $D_{ii} = \sum_j w_{ij}$) and W is the similarity matrix.

Spectral Clustering

Let's compute D and $D^{-\frac{1}{2}}$:

$$D = \begin{bmatrix} 3.547 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 3.626 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 3.535 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 3.264 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 2.806 \end{bmatrix}$$

$$D^{-\frac{1}{2}} = \begin{bmatrix} 0.531 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.525 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.532 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.554 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.597 \end{bmatrix}$$

Spectral Clustering

The identity matrix $I = I(5)$ be

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The Laplacian Matrix becomes:

$$L = I - D^{-1/2} W D^{-1/2}$$

$$L = \begin{bmatrix} 0.718 & -0.234 & -0.202 & -0.144 & -0.158 \\ -0.234 & 0.724 & -0.269 & -0.147 & -0.099 \\ -0.202 & -0.269 & 0.717 & -0.166 & -0.092 \\ -0.144 & -0.147 & -0.166 & 0.694 & -0.232 \\ -0.158 & -0.099 & -0.092 & -0.232 & 0.644 \end{bmatrix}$$

Spectral Clustering

3. Eigenvalue Decomposition:
We compute the eigenvalues and eigenvectors of the Laplacian matrix L. Let's say we find the following eigenvalues (in ascending order) and their corresponding eigenvectors:

Eigenvalue 1: 0.000, Eigenvector: [-0.460, -0.465, -0.459, -0.441, -0.409]
Eigenvalue 2: 0.668, Eigenvector: [0.184, 0.412, 0.394, -0.385, -0.702]
Eigenvalue 3: 0.878, Eigenvector: [-0.596, -0.013, 0.298, 0.655, -0.355]
Eigenvalue 4: 0.954, Eigenvector: [-0.582, 0.191, 0.434, -0.473, 0.462]
Eigenvalue 5: 0.997, Eigenvector: [0.246, -0.760, 0.597, -0.069, -0.008]

Spectral Clustering

4. Dimensionality Reduction:

We select the eigenvectors corresponding to the smallest eigenvalues to embed the data into a lower-dimensional space. In practice, we often choose the eigenvectors corresponding to the smallest non-zero eigenvalues. Let's say we choose the eigenvectors corresponding to the first and second smallest eigenvalues:

$$\begin{bmatrix} 0.184 & -0.460 \\ 0.412 & -0.465 \\ 0.394 & -0.459 \\ -0.385 & -0.441 \\ -0.702 & -0.409 \end{bmatrix}$$

Spectral Clustering

5. Clustering:
Finally, we perform clustering on the lower-dimensional embedding. Let's say we use k-means clustering with $k = 2$.

Applying k-means clustering to the two-dimensional embedding, we find two clusters:

- Cluster 1: A, B, C
- Cluster 2: D, E

So, in this example, spectral clustering grouped data points A, B, and C into one cluster and data points D and E into another cluster.

Spectral Clustering

```
> DC
       X1     X2
A   0.184 -0.460
B   0.412 -0.465
C   0.394 -0.459
D -0.385 -0.441
E -0.702 -0.409
```

Spectral Clustering

```
> set.seed(199999)
> km=kmeans(DC,2,iter.max = 1000, nstart = 1,
      algorithm = c("MacQueen"))
> km
K-means clustering with 2 clusters of sizes 3, 2
Cluster means:
        X1          X2
1   0.3300 -0.4613333
2 -0.5435 -0.4250000
Clustering vector:
A B C D E
1 1 1 2 2
Within cluster sum of squares by cluster:
[1] 0.03215667 0.05075650
 (between_SS / total_SS =  91.7 %)
```

Spectral Clustering

In what situation can use Spectral Clustering?

Spectral clustering is particularly useful in several specific situations where other clustering methods might struggle. Here are some common scenarios where spectral clustering excels:

- Non-Linearly Separable Data: When data points cannot be separated by a straight line or hyperplane (e.g., concentric circles or spirals). For Example, Clustering images of handwritten digits where the digit shapes vary significantly.
- Graph-Based Data: When data naturally forms a graph structure, such as social networks, where nodes represent entities and edges represent relationships. Community detection in social networks, where individuals form distinct groups based on their connections.
- Complex Data with Manifold Structures: When data lies on a low-dimensional manifold embedded in a higher-dimensional space. Clustering high-dimensional sensory data that lies on a lower-dimensional manifold, like facial images under varying lighting conditions.

Spectral Clustering

In what situation can use Spectral Clustering?.....

- Small to Medium-Sized Datasets: When the dataset size is not extremely large (typically up to a few thousand samples), as spectral clustering involves computing the eigenvalues and eigenvectors of the affinity matrix, which can be computationally intensive. Clustering a few thousand samples of gene expression data in bioinformatics.
- Clustering with Connectivity Constraints: When you need to incorporate connectivity information in the clustering process. Image segmentation, where spatial connectivity between pixels is important.
- Clustering with Arbitrary Shape Clusters: When clusters are not necessarily spherical or of uniform size. Clustering geographical data where clusters could be irregularly shaped regions.

Spectral Clustering

Key Strengths of Spectral Clustering

- Flexibility with Cluster Shapes: Unlike K-means, which assume spherical clusters, spectral clustering can handle clusters of arbitrary shapes.
- Effective with Non-Convex Clusters: It performs well with data that forms non-convex clusters, where traditional methods like K-means fail.
- Graph-Theoretic Approach: Utilizes the properties of eigenvectors of a similarity matrix derived from the data, making it powerful for graph-based applications.
- Manifold Learning: Can uncover the underlying manifold in high-dimensional data, making it useful for dimensionality reduction and data visualization.

Spectral Clustering

When to Avoid Spectral Clustering

- Very Large Datasets: Spectral clustering involves operations like computing the eigenvectors of the Laplacian matrix, which can be computationally expensive for very large datasets.
- Scalability Issues: If scalability is a concern, approximate methods or other clustering techniques like K-means or DBSCAN might be more suitable.

Spectral Clustering

Practical Examples of Spectral Clustering

- Image Segmentation: Grouping pixels in an image into regions or segments based on color, texture, or intensity.
- Social Network Analysis: Detecting communities within social networks based on the structure of connections.
- Document Clustering: Grouping documents based on content similarity, useful in information retrieval and text mining.
- Sensor Networks: Clustering sensor nodes based on the data they collect and their geographic or functional similarities.

By leveraging the graph structure and eigenvalues, spectral clustering can reveal intricate patterns in data, making it a powerful tool for various advanced clustering tasks.

Spectral Clustering

Spectral Clustering for Moon Data.

Example Generate moon data and apply spectral clustering to cluster data. Plot the data for the final cluster solution.

Spectral Clustering

Spectral Clustering for Moon data.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.cluster import SpectralClustering
# Generate some sample data (moons dataset)
X, y = make_moons(n_samples=300, noise=0.1, random_state=42)
# Plot the original data
plt.figure(figsize=(6, 6))
plt.scatter(X[:, 0],X[:, 1],c='b',marker='o',edgecolors='k')
plt.title("Original 'Two Moons' Data")
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid(True)
plt.savefig("D:/Origiunal Moon Data.jpg", dpi=700 )
plt.show()
```

Spectral Clustering

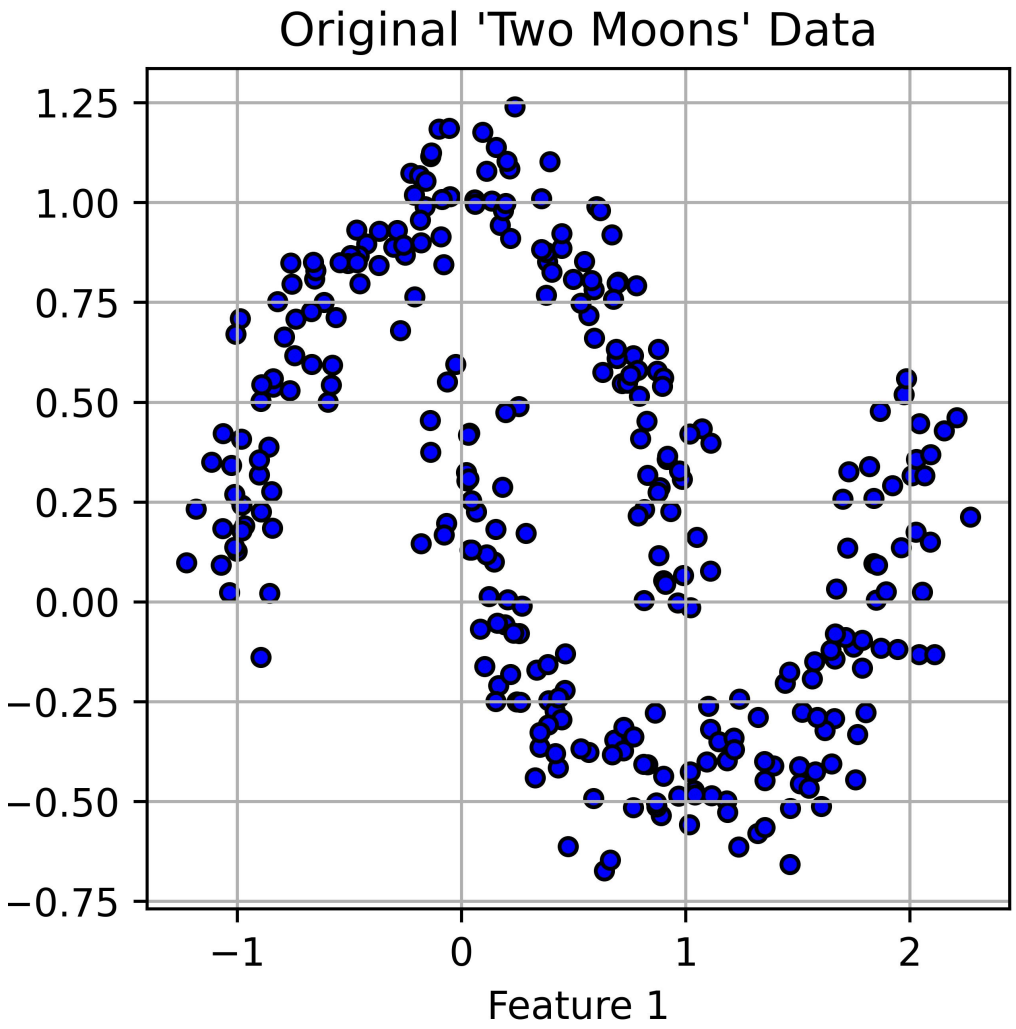


Figure: Schematic plot for Moon data.

Spectral Clustering

```
# Spectral Clustering
spectral_clustering = SpectralClustering(n_clusters=2,
        n_neighbors=10, affinity='nearest_neighbors',
                                        random_state=42)
y_pred = spectral_clustering.fit_predict(X)

# Plot the clustered data
plt.figure(figsize=(4, 4))
plt.scatter(X[:, 0], X[:, 1], c=y_pred,s=20, marker='o',
                edgecolors='k')
plt.title("Spectral Clustering on 'Two Moons' Data")
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid(True)
plt.show()
```

Spectral Clustering

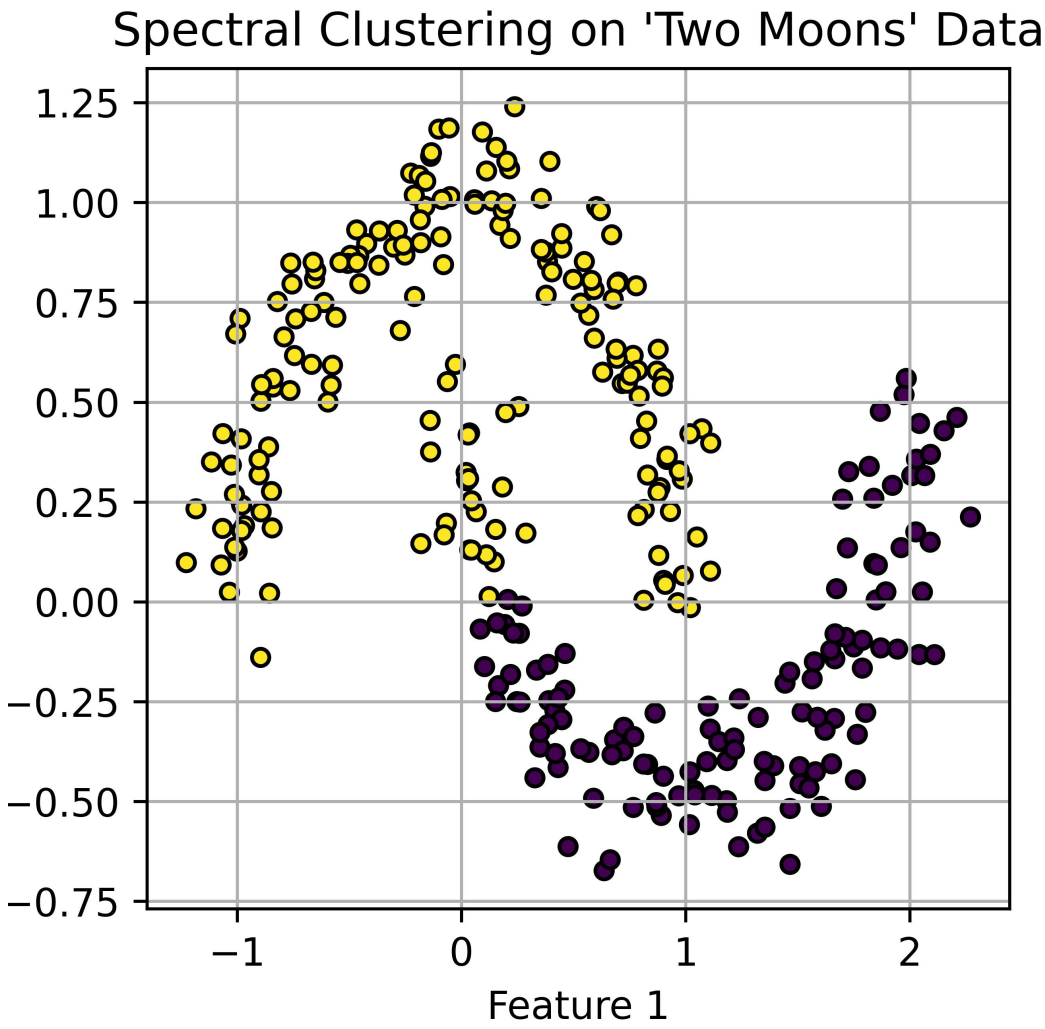


Figure: Schematic plot for spectral clustering of Moon data.

Optimal number of Cluster for Spectral Clustering

Determining the optimal number of clusters in spectral clustering involves several steps. Here is a structured approach to achieve this:

1. Data Preparation: Ensure that your data is in a suitable format, typically an $n \times m$ matrix where n is the number of samples and m is the number of features.
2. Construct the Similarity Graph: Create a similarity graph where nodes represent data points and edges represent the similarity between them. This can be done using various methods such as:
   k-nearest neighbors (k-NN) graph: Connect each node to its k nearest neighbors.
   ϵ-neighborhood graph: Connect nodes if their distance is less than a threshold ϵ.
   Fully connected graph: Use a similarity measure (e.g., Gaussian similarity) for all pairs of nodes.

Optimal number of Cluster for Spectral Clustering

3. Construct the Similarity Matrix: Construct a similarity matrix S where S_{ij} represents the similarity between nodes i and j. Common choices include:
   Gaussian (RBF) Kernel:
   $$S_{ij} = exp\left[- \frac{||(x_i - x_j)||^2}{2\sigma^2} \right] = exp\left[- \gamma||(x_i - x_j)||^2 \right], \text{ where } \gamma = \frac{1}{2\sigma^2}$$
   Binary Connectivity: $S_{ij} = 1$ if nodes i and j are connected, otherwise 0.
4. Construct the Laplacian Matrix: Compute the Laplacian matrix L. There are different types of Laplacians:
   Unnormalized Laplacian: $L = D - S$,
   Normalized Laplacian: $L_{sym} = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$ or $L_{rw} = I - D^{-1} S$.
   Here, D is the degree matrix, a diagonal matrix where $D_{ii} = \sum_j S_{ij}$.
5. Compute Eigenvalues and Eigenvectors: Compute the eigenvalues and corresponding eigenvectors of the Laplacian matrix. For normalized Laplacian, solve $L_{sym} v = \lambda v$ or $L_{rw} v = \lambda v$.

Optimal number of Cluster for Spectral Clustering

6. Determine the Optimal Number of Clusters: To determine the optimal number of clusters, analyze the eigenvalues and eigenvectors: Eigenvalue Gap (Spectral Gap): Identify the largest gap between consecutive eigenvalues (e.g., the gap between λ_k and λ_{k+1}). The position k of the largest gap suggests the optimal number of clusters. Eigengap Heuristic: Plot the eigenvalues in ascending order and look for a significant drop.
7. Cluster the Data: Use the first k eigenvectors (corresponding to the smallest k eigenvalues) to form a new feature matrix U. Each row of U represents a data point in the reduced k-dimensional space.
8. Apply k-means Clustering: Apply k-means clustering on the rows of the matrix U to obtain the final clusters.
9. Validate and Interpret Clusters: Validate the clustering results using internal evaluation metrics (e.g., silhouette score, Davies-Bouldin index) or external validation (if ground truth labels are available).

Optimal number of Cluster for Spectral Clustering

Summary of Steps:

- Data Preparation: Prepare the data matrix.
- Similarity Graph: Construct a similarity graph.
- Similarity Matrix: Create the similarity matrix S.
- Laplacian Matrix: Compute the Laplacian matrix L.
- Eigenvalues and Eigenvectors: Compute the eigenvalues and eigenvectors of L.
- Optimal Clusters: Determine the optimal number of clusters using the eigenvalue gap or eigengap heuristic.
- New Feature Matrix: Form a new feature matrix using the first k eigenvectors.
- k-means Clustering: Apply k-means clustering on the new feature matrix.
- Validation: Validate and interpret the clustering results.

By following these steps, you can effectively determine the optimal number of clusters using spectral clustering.

Optimal number of Cluster for Spectral Clustering

Example Generate blobs of points of size 300 and apply spectral clustering to obtain the cluster solution. Also, before applying the cluster solution find the optimal number of clusters.

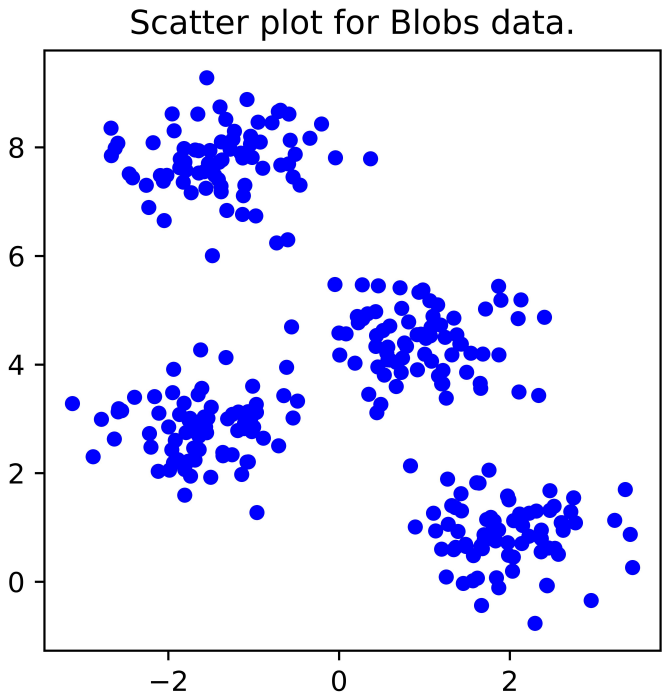
Optimal number of Cluster for Spectral Clustering

Python Code for optimal number of Cluster for Spectral Clustering

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import SpectralClustering, KMeans
from sklearn.metrics import pairwise_kernels
from sklearn.datasets import make_blobs
from scipy.sparse.linalg import eigsh
from sklearn.metrics import silhouette_score

# Generate synthetic data
X, y_true = make_blobs(n_samples=300, centers=4,
                       cluster_std=0.60, random_state=0)
# Construct the similarity matrix using RBF kernel
sigma = 1.0
S = pairwise_kernels(X, metric='rbf',
                gamma=1.0 / (2 * sigma ** 2))
```

Optimal number of Cluster for Spectral Clustering

```
# Plot the clustered data
plt.figure(figsize=(4, 4))
plt.scatter(X[:, 0], X[:, 1],  s=20, c='blue',cmap='viridis')
plt.title('Scatter plot for Blobs data.')
plt.savefig('D:/Plot for Blobs data.jpg',dpi=1200)
plt.show()
```



Optimal number of Cluster for Spectral Clustering

```
# Compute the Degree Matrix
D = np.diag(np.sum(S, axis=0))
# Compute the Laplacian Matrix
L = D - S

# Normalized Laplacian (Symmetric)
D_inv_sqrt = np.diag(1.0 / np.sqrt(np.diag(D)))
L_sym = np.identity(D.shape[0]) - D_inv_sqrt @ S @ D_inv_sqrt



# Compute the first k eigenvalues and eigenvectors
# of the normalized Laplacian
k = 10  # Number of eigenvalues or vectors to compute
# ='SM' : Smallest (in magnitude) eigenvalues.
eigenvalues, eigenvectors = eigsh(L_sym, k=k, which='SM')
```

Optimal number of Cluster for Spectral Clustering

```
# Sort eigenvalues to find the largest gap
sorted_eigenvalues = np.sort(eigenvalues)
gaps = np.diff(sorted_eigenvalues)
# Find the index of the largest gap
optimal_k = np.argmax(gaps) + 1
```

Optimal number of Cluster for Spectral Clustering

```
# Plot the eigenvalues and gaps to visualize
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(range(1, k+1), sorted_eigenvalues, marker='o',c='blue
plt.title('Eigenvalues')
plt.xlabel('Index')
plt.ylabel('Eigenvalue')

plt.subplot(1, 2, 2)
plt.plot(range(1, k), gaps, marker='o',c='blue')
plt.title('Eigengaps')
plt.xlabel('Index')
plt.ylabel('Gap')
plt.axvline(x=optimal_k-1, color='r', linestyle='--')
plt.show()
```

Optimal number of Cluster for Spectral Clustering

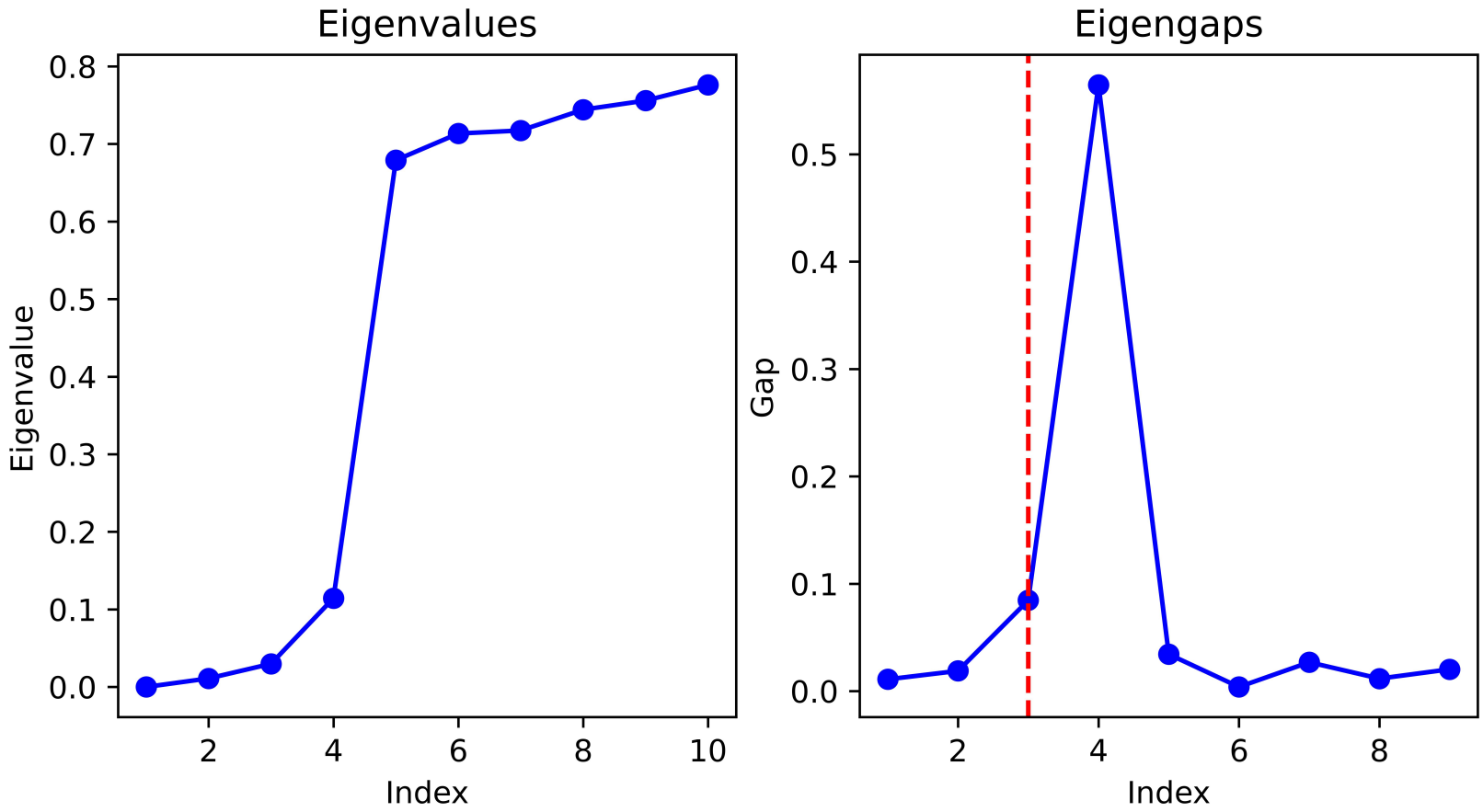


Figure: Schematic plot for Eigenvalues.

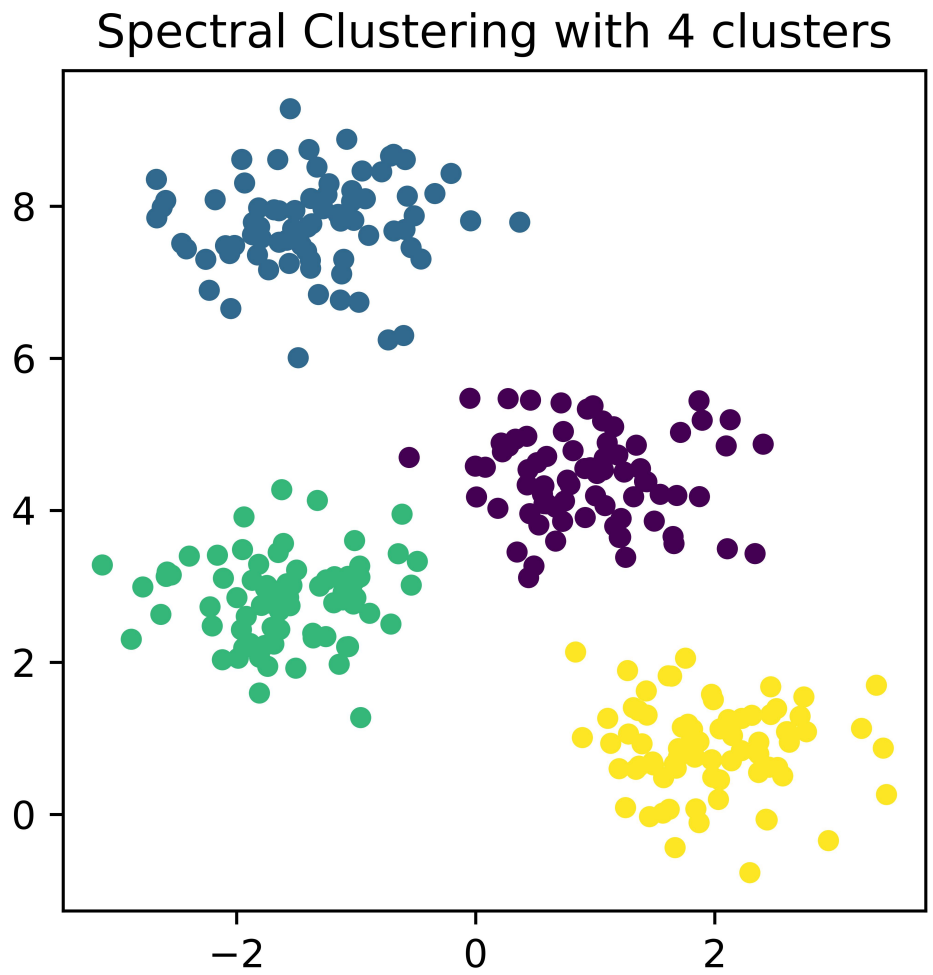
Optimal number of Cluster for Spectral Clustering

```
print(f"The optimal number of clusters is {optimal_k}")
The optimal number of clusters is 4

#Apply Spectral Clustering with the optimal number of clusters
sc = SpectralClustering(n_clusters=optimal_k, affinity='rbf',
                        gamma=1.0 / (2 * sigma ** 2), random_state=0)
labels = sc.fit_predict(X)

# Validate using silhouette score
score = silhouette_score(X, labels)
print(f"Silhouette Score: {score}")
Silhouette Score: 0.6819938690643478
```

Optimal number of Cluster for Spectral Clustering

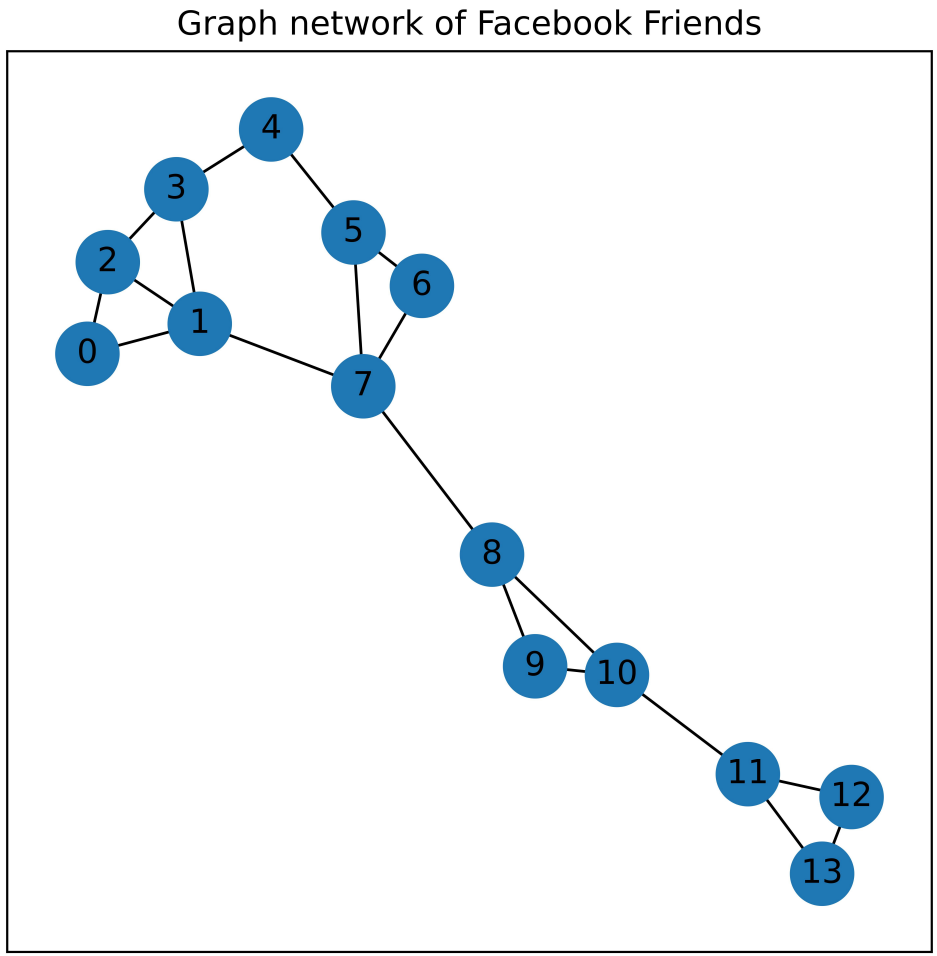
```
# Plot the clustered data
plt.figure(figsize=(4, 4))
plt.scatter(X[:, 0], X[:, 1], c=labels, s=20, cmap='viridis')
plt.title(f'Spectral Clustering with {optimal_k} clusters')
plt.show()
```

Optimal number of Cluster for Spectral Clustering



Spectral Clustering for Facebook Friends

Example: Apply spectral clustering for the following graph. Use Python to solve the problem.



Spectral Clustering for Facebook Friends

```
# Spectal Clustering for Facebook Friends
import scipy
#pip install --upgrade scipy
from scipy.sparse import coo_matrix
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from sklearn.cluster import SpectralClustering
from sklearn.metrics import pairwise_kernels
```

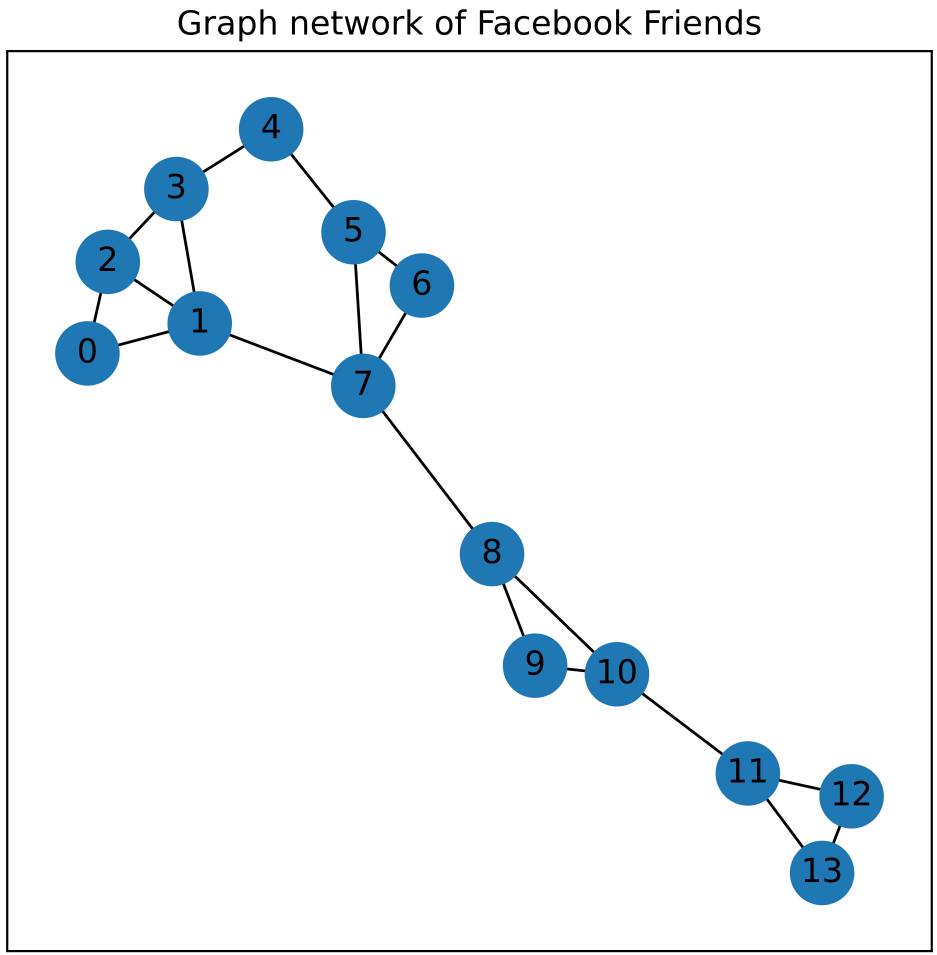
Spectral Clustering for Facebook Friends

```
# Example data: adjacency list
edges = [
    (0, 1), (0, 2), (1, 2), (1, 3),(1,7),
    (2, 3), (3, 4), (4, 5), (5, 6),
    (5, 7), (6, 7), (7, 8), (8, 9),
    (8, 10), (9, 10),(10, 11), (11, 12),
    (11, 13), (12, 13)
]

# Create the graph
G = nx.Graph()
G.add_edges_from(edges)
```

Spectral Clustering for Facebook Friends

```
# Draw the graph
posg = nx.spring_layout(G)  # Layout for visualization
plt.figure(figsize=(6, 6))
nx.draw_networkx(G, posg,  with_labels=True,
                 node_size=500, cmap=plt.cm.tab10)
plt.title('Graph network of Facebook Friends')
plt.savefig('D:/Graph network Facebook Friends.jpg',dpi=700)
plt.show()
```

Spectral Clustering for Facebook Friends



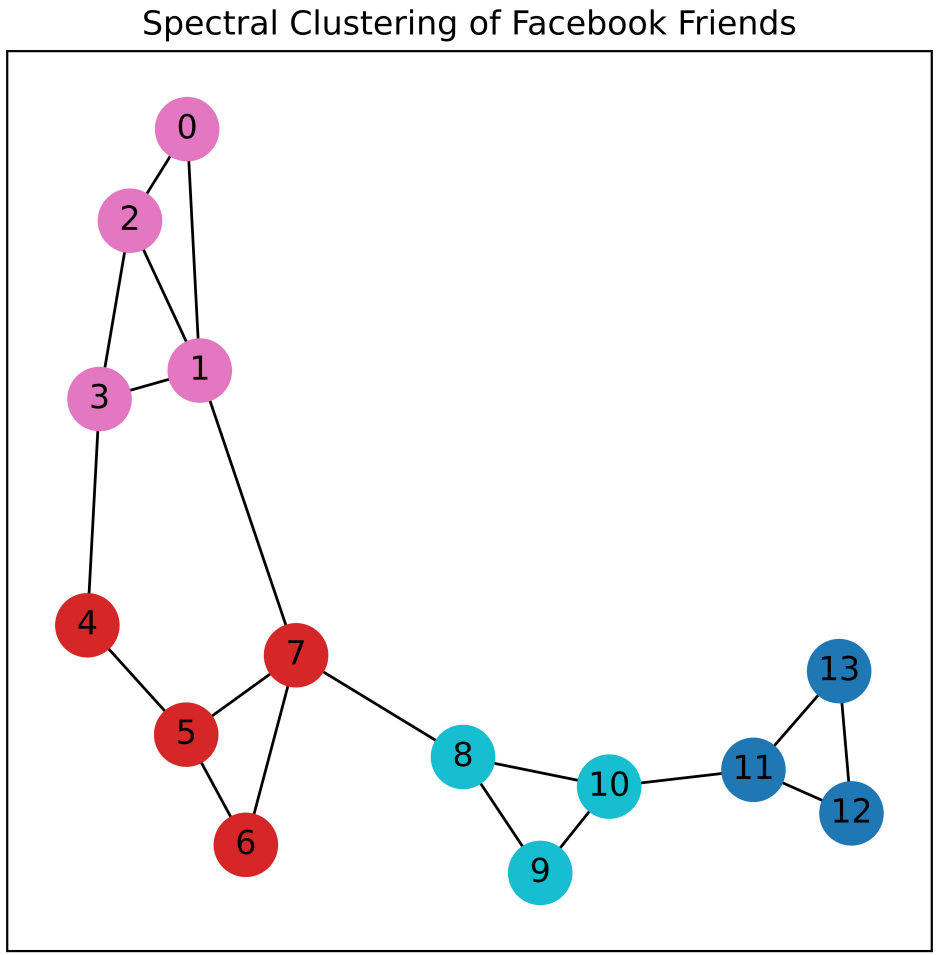
Spectral Clustering for Facebook Friends

```
# Create adjacency matrix
A = nx.adjacency_matrix(G).todense()

# Apply spectral clustering
n_clusters = 4  # Number of clusters
sc = SpectralClustering(n_clusters=n_clusters,
                        affinity='precomputed',random_state=0)
labels = sc.fit_predict(A)

# Assign colors to nodes based on cluster labels
colors=[plt.cm.tab10(i/float(n_clusters-1)) for i in labels]
```

Spectral Clustering for Facebook Friends

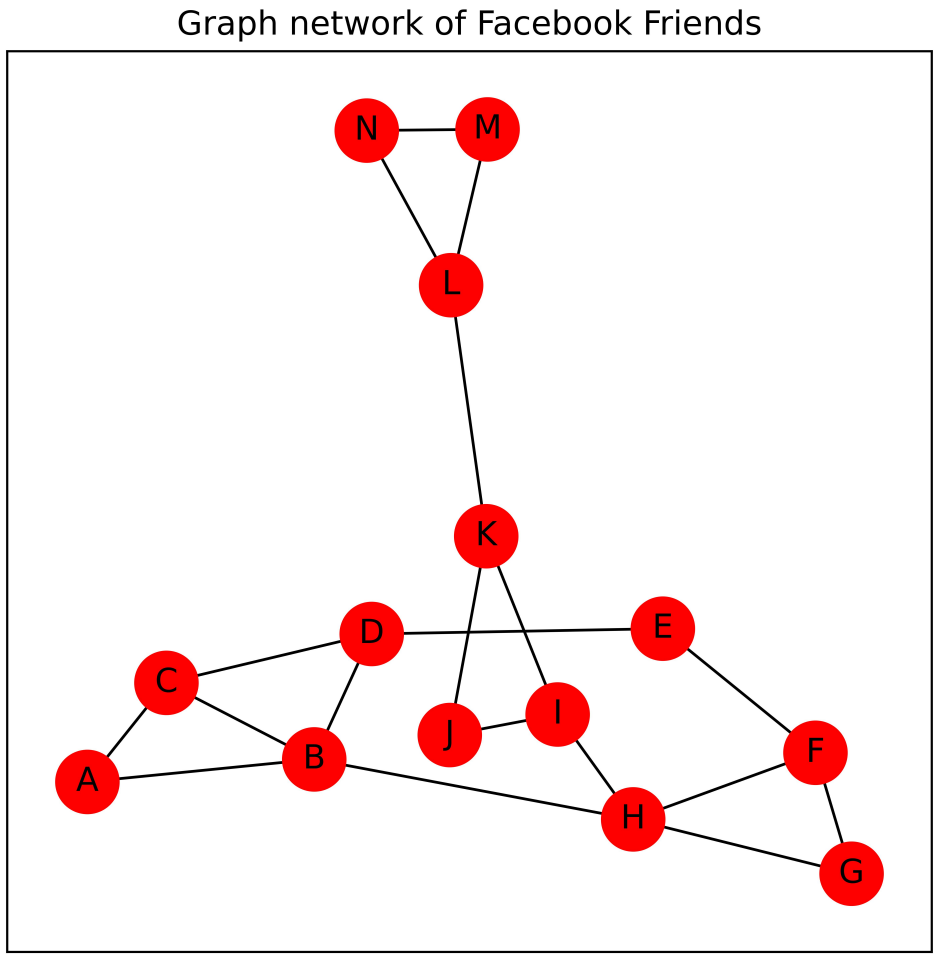
```
# Draw the graph
pos = nx.spring_layout(G)  # Layout for visualization
plt.figure(figsize=(6, 6))
nx.draw_networkx(G, pos, node_color=colors, with_labels=True,
                 node_size=500, cmap=plt.cm.tab10)
plt.title('Spectral Clustering of Facebook Friends')
#plt.savefig('D:/Spectral Clustering Facebook Friends.jpg',
            dpi=1200)
plt.show()
```

Spectral Clustering for Facebook Friends



Spectral Clustering for Facebook Friends

Example: Apply spectral clustering for the following graph. Use Python to solve the problem.



Spectral Clustering for Facebook Friends

```
# Spectal Clustering for Facebook Friends
import scipy
#pip install --upgrade scipy
from scipy.sparse import coo_matrix
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from sklearn.cluster import SpectralClustering
from sklearn.metrics import pairwise_kernels
```

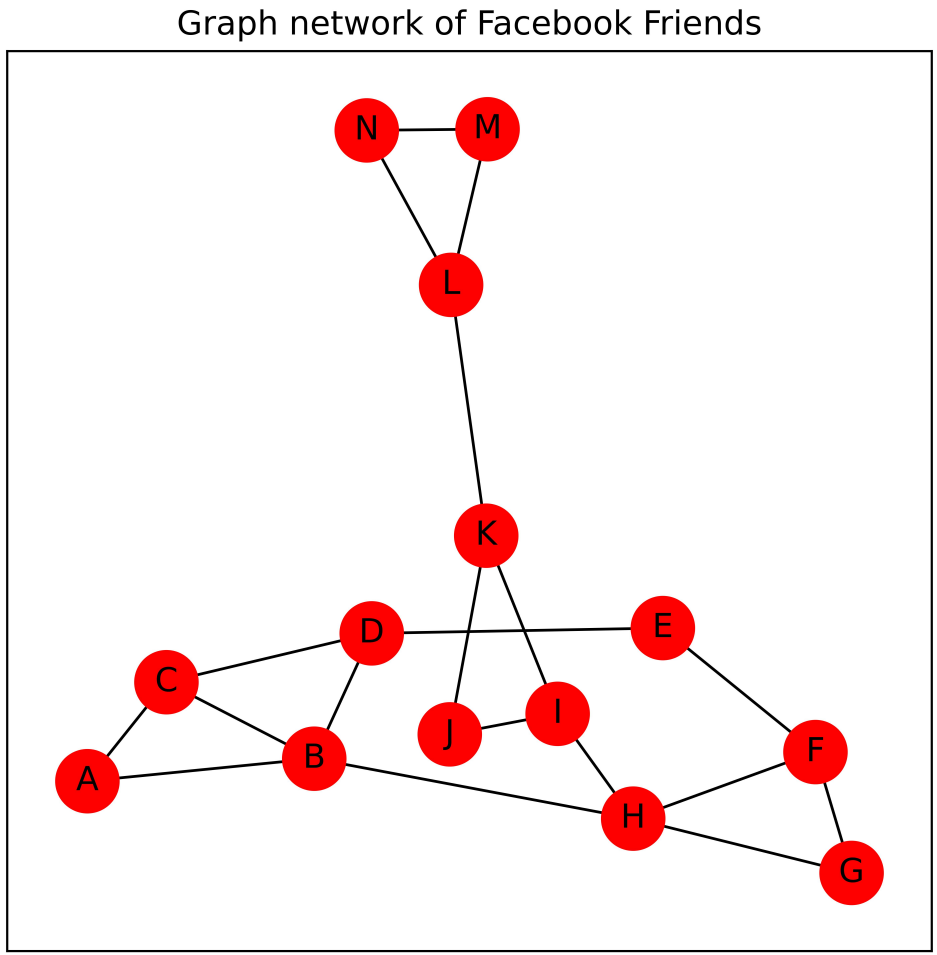
Spectral Clustering for Facebook Friends

```
# Example data: adjacency list
edges = [
    ('A', 'B'), ('A', 'C'), ('B', 'C'), ('B', 'D'),('B','H'),
    ('C', 'D'), ('D', 'E'), ('E', 'F'), ('F', 'G'),
    ('F', 'H'), ('G', 'H'), ('H', 'I'), ('I', 'J'),
    ('I', 'K'), ('J', 'K'),('K','L'), ('L', 'M'),
    ('L', 'N'), ('M', 'N') ]
```

Spectral Clustering for Facebook Friends

```
# Create the graph
G = nx.Graph()
G.add_edges_from(edges)
# Draw the graph
posg = nx.spring_layout(G)  # Layout for visualization
plt.figure(figsize=(6, 6))
nx.draw_networkx(G, posg,  with_labels=True,
                 node_size=500, cmap=plt.cm.tab10)
plt.title('Graph network of Facebook Friends')
plt.show()
```

Spectral Clustering for Facebook Friends



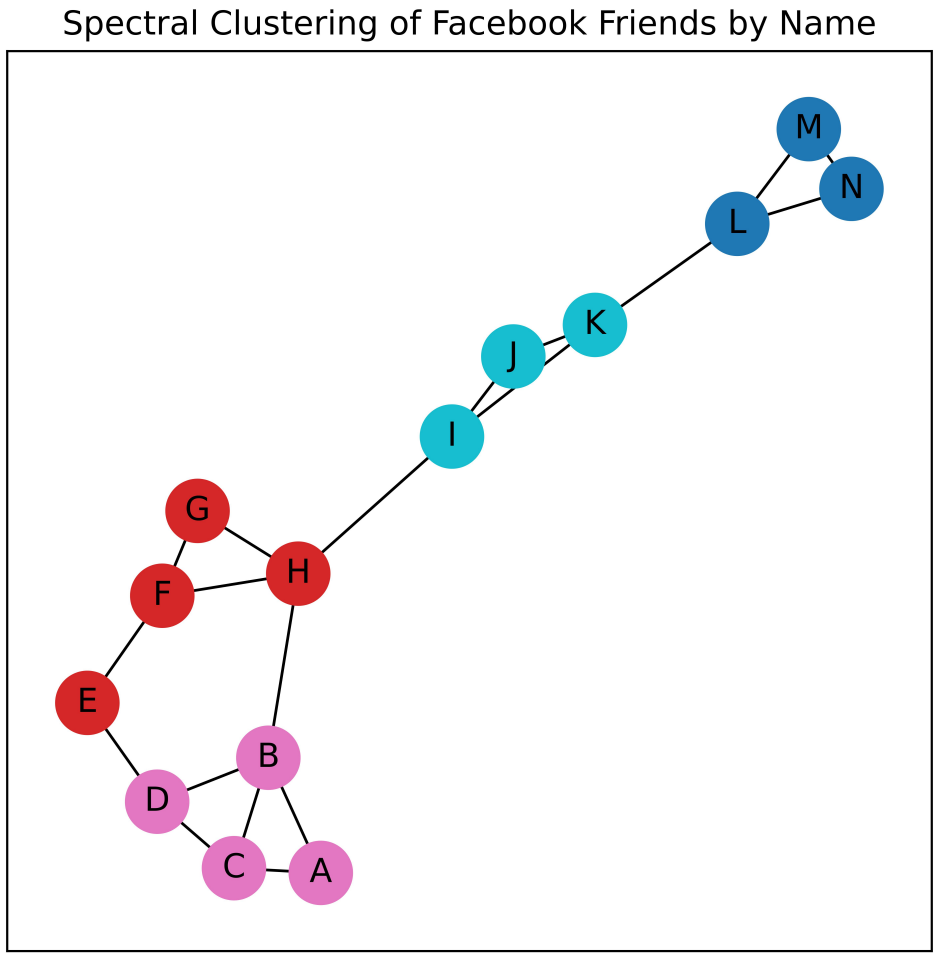
Spectral Clustering for Facebook Friends

```
# Create adjacency matrix
A = nx.adjacency_matrix(G).todense()

# Apply spectral clustering
n_clusters = 4  # Number of clusters
sc = SpectralClustering(n_clusters=n_clusters,
                        affinity='precomputed',random_state=0)
labels = sc.fit_predict(A)

# Assign colors to nodes based on cluster labels
colors=[plt.cm.tab10(i/float(n_clusters-1)) for i in labels]
```

Spectral Clustering for Facebook Friends

```
# Draw the graph
pos = nx.spring_layout(G)  # Layout for visualization
plt.figure(figsize=(6, 6))
nx.draw_networkx(G, pos, node_color=colors, with_labels=True,
                 node_size=500, cmap=plt.cm.tab10)
plt.title('Spectral Clustering of Facebook Friends')
#plt.savefig('D:/Spectral Clustering Facebook Friends.jpg',
             dpi=1200)
plt.show()
```

Spectral Clustering for Facebook Friends



***** - /// - There are no End .. - /// - *****

****

****