National University of Sciences and Technology
School of Electrical Engineering and Computer Science
Department of Computing

CS 471 Machine Learning

Fall 2024

Assignment 3

Spectral Clustering

Practical Application in Identifying Social Networks Through a Graph-Based Dataset

Announcement Date: 8th Dec 2024

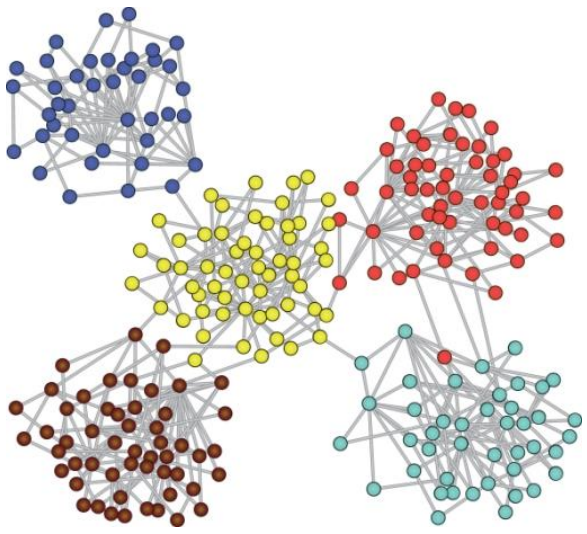
Due Date: 25th Dec 2024 at 11:59 PM (on LMS)

Instructor: Prof. Dr. Muhammad Moazam Fraz

Table of Contents

1. Introduction

Clustering is a fundamental technique in machine learning, often used to discover patterns and structure in data without predefined labels. In this assignment, we explore **spectral clustering**, a powerful approach that uses graph theory to group data points based on their relationships and similarities. Spectral clustering is particularly effective for complex datasets where traditional methods fail to capture underlying patterns, especially when the data can be represented as a graph.



The dataset for this assignment comes from the **Facebook Social Circles Dataset**, which represents the interactions and friendships within a social network. By applying spectral clustering, we aim to uncover hidden communities or social circles within this network, providing information regarding how individuals are connected and interact within their social ecosystem. This exercise offers you a unique opportunity to explore graph-based machine learning, practice clustering techniques, and gain a deeper understanding of how relationships in real-world networks can be analyzed and visualized.

1.1 Problem Statement

In the digital era, social networks play a crucial role in connecting people, facilitating interactions, and forming communities. Understanding the structure of these networks is key to uncovering meaningful info, such as identifying tightly-knit groups or influential individuals. The **Facebook Social Circles Dataset** provides a graph-based representation of a social network, where nodes correspond to individuals and edges represent friendships or interactions between them.

Your task is to apply **spectral clustering** to this dataset to identify distinct social circles within the network. By using the power of graph theory and spectral analysis, you will group individuals into clusters based on their connectivity patterns. This will involve constructing a similarity graph, computing the Laplacian matrix, and using eigenvector-based representations to perform clustering. The goal is to analyze the clusters to reveal underlying community structures and interpret the social dynamics they represent. Through this exercise, you will be enhancing your understanding of graph-based learning and the practical applications of spectral clustering in real-world problems.

1.2 Purpose and Objectives

This assignment bridges theoretical concepts in graph theory with practical machine learning techniques, emphasizing the importance of data-driven approaches in understanding social systems.

The purpose of this assignment is to:

- Learn the **theoretical foundations** of spectral clustering and how it utilizes **graph representations** and **eigenvector computations** to partition data.
- Construct a **similarity graph** from the given dataset, compute the **Laplacian matrix**, and perform **eigenvalue decomposition** to enable clustering.
- **Identify** and **analyze** social circles within the network, interpreting the relationships and structures revealed by the clusters.
- Gain proficiency in **graph data preprocessing**, implementing spectral clustering algorithms, and **evaluating clustering results** using visualizations and metrics.
- Understand the **applications of spectral clustering** in fields of social network analysis, recommendation systems, and biological networks, connecting theoretical knowledge with impactful solutions.

By the end of this assignment, you will not only deepen your understanding of machine learning techniques for graph-based data but also appreciate their versatility in addressing complex, unstructured problems across various domains.

1.3 Assignment Structure

This assignment is structured to guide you through the process of applying spectral clustering to the Facebook Social Circles Dataset. It is divided into 4 major sections, each focusing on key steps required to understand, implement, and evaluate spectral clustering for community detection in social networks.

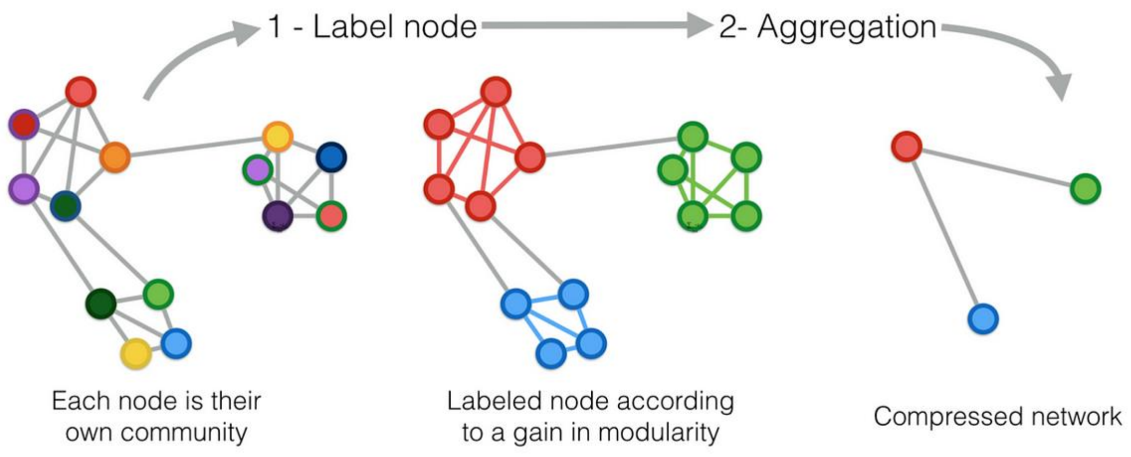
- **Understanding Spectral Clustering -** This section introduces the **concepts** behind spectral clustering, including its connection to **graph theory**, the role of the **Laplacian matrix**, and the use of **eigenvectors** for dimensionality reduction.
- **Dataset Exploration and Graph Construction -** Explore the Facebook Social Circles Dataset, **preprocess** the data, and **construct a similarity graph**. This includes visualizing the graph structure and **defining meaningful similarity measures**.
- **Applying Spectral Clustering -** This section focuses on the core algorithm, including computing the **Laplacian matrix**, finding its **eigenvectors**, and **applying clustering** (k-means) to identify social circles within the network.
- **Evaluating Clustering Results -** Analyze the **quality of your clusters** using visualization techniques and **appropriate evaluation metrics**.
- **Discussion and Conclusion -** Summarize your **key findings**, discuss different factors contributing to the formation of clusters in this data, **reflect** on the performance of your model, and **explain** what you learnt during this assignment.

2. Understanding Spectral Clustering

2.1 Graph-Based Representation

Spectral clustering operates on a graph representation of the data, making it particularly well-suited for datasets where relationships between data points are as important as the points themselves. In this approach, the dataset is represented as a graph $G=(V,E)$ where:

- V represents the set of nodes (data points).
- E represents the edges, which denote the relationships or similarities between nodes.
- The strength of these relationships is captured using edge weights, stored in an adjacency matrix A.


1 - Label node
2- Aggregation
Each node is their own community
Labeled node according to a gain in modularity
Compressed network

The adjacency matrix A is a symmetric matrix where each entry A_{ij} quantifies the similarity between node *i* and node *j*. Common similarity measures include the Gaussian kernel or *k*-nearest neighbors.

This graph representation transforms the clustering problem into one of partitioning the graph into subgraphs (clusters) such that nodes within the same subgraph are highly connected (similar), while connections between subgraphs are minimized. Spectral clustering achieves this by using the **Laplacian matrix** derived from the adjacency matrix, capturing the graph's connectivity and structural properties.

By focusing on this graph-based representation, spectral clustering provides flexibility and robustness, making it effective for non-linearly separable data and datasets with complex structures, like those in a social network.

2.2 Steps in Spectral Clustering

Spectral clustering transforms the problem of grouping data points into clusters into a graph partitioning problem, using the eigenstructure of the graph's Laplacian matrix. The process involves the following key steps:

- **Construct the Similarity Graph**
  Represent the dataset as a graph $G=(V,E)$, where nodes correspond to data points and edges reflect the similarity between points. The adjacency matrix A encodes these similarities using measures like the Gaussian kernel or k-nearest neighbors.

- **Compute the Laplacian Matrix**
  From the adjacency matrix, compute the graph Laplacian L. There are two common forms:
  - Unnormalized Laplacian: $L = D - A$, where D is the degree matrix ($D_{ii} = \sum_j A_{ij}$).
  - Normalized Laplacian: $L_{\text{sym}} = I - D^{-1/2} A D^{-1/2}$ or $L_{\text{rw}} = I - D^{-1} A$.
- **Calculate Eigenvectors**
  Perform eigenvalue decomposition on the Laplacian matrix and select the eigenvectors corresponding to the smallest k eigenvalues (where k is the number of clusters). These eigenvectors form a reduced, low-dimensional representation of the graph.
- **Cluster in Reduced Space**
  Treat the rows of the eigenvector matrix as features and apply a traditional clustering algorithm (e.g., k-means) to group the data points into k clusters.
- **Interpret Results**
  Map the clusters back to the original graph to analyze and visualize the identified partitions. Each cluster represents a tightly connected subgraph within the larger network.

These steps combine to partition the graph into meaningful clusters, making spectral clustering a powerful tool for analyzing complex, non-linearly separable data.

2.3 Challenges

While spectral clustering is a versatile and powerful technique, its effectiveness depends on careful handling of several challenges and considerations.

The similarity graph heavily influences the results. Selecting an appropriate similarity measure (e.g., Gaussian kernel or k-nearest neighbors) and tuning its parameters (e.g., σ for Gaussian kernel) are crucial for accurately capturing relationships in the data. Spectral clustering involves eigenvalue decomposition of the Laplacian matrix, which can be computationally expensive for large datasets. Efficient implementations or approximations may be necessary for scalability. Determining the optimal number of clusters is often non-trivial and may require domain knowledge or evaluation metrics like the silhouette score or the eigengap heuristic.

Constructing the similarity graph, particularly deciding on the graph type (e.g., fully connected vs. k-nearest neighbors), impacts the clustering outcome. Sparse graphs (like k-NN) are often preferred to reduce noise and computational load. Parameters such as the number of neighbors in k-NN or the scaling factor in the Gaussian kernel directly affect the clustering results. Hyperparameter tuning is essential to achieve meaningful clusters. Unlike centroid-based clustering methods, spectral clustering results are tied to the graph structure, which may make interpreting the clusters less intuitive for some applications.

3. Dataset Exploration and Graph Construction

3.1 Understanding the Dataset

The **Facebook Social Circles Dataset** provides a rich graph-based representation of social connections, making it ideal for exploring community detection through spectral clustering. It contains information about the ego networks of individuals, their friends, and the features associated with each user. The dataset is anonymized to preserve privacy, with user attributes and connections represented as abstract features and edges.

Key Components of the Dataset

Nodes and Edges

Nodes: Represent Facebook users (4039 users in total).

Edges: Represent friendships between users (88234 edges in total). These are undirected for Facebook networks.

Ego Networks

An ego network consists of a central "ego" user and all their direct connections ("friends"). The ego nodes themselves are not explicitly included in the edges but are assumed to connect to all their listed friends.

Circles

Each ego node has associated "circles" or friend groups, which capture subgroups of friends within their network. These provide ground truth for validating clustering results.

Features

Users have anonymized binary feature vectors indicating specific attributes (e.g., political affiliation or interests), but the actual interpretations of these features are hidden.

Graph Statistics

Nodes in the largest connected component (WCC/SCC): 4039

Edges in the largest connected component: 88234

Average clustering coefficient: 0.6055 (indicating a high tendency for users' friends to be friends themselves).

Number of triangles: 1,612,010 (indicative of dense local connectivity).

Diameter: 8 (longest shortest path in the network).

90-percentile effective diameter: 4.7.

Files and Their Contents

nodeId.edges - Lists edges (friendship connections) for the ego network of a specific node.

nodeId.circles - Describes the social circles (groups) for the ego node, with each line corresponding to a specific circle.

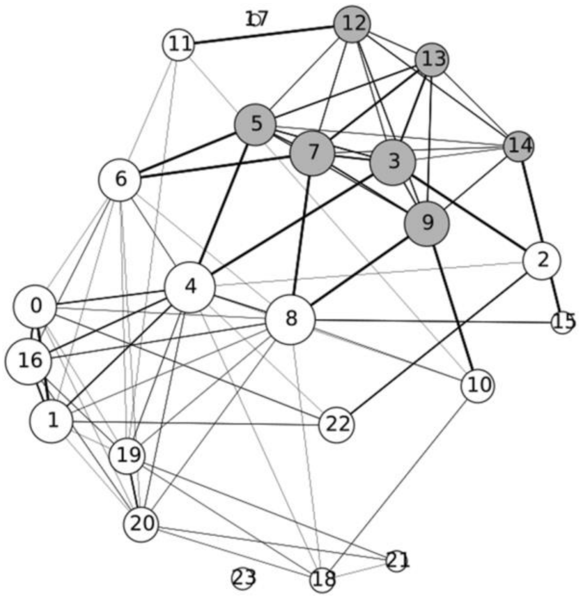
nodeId.feat - Binary feature vectors for the nodes in the network.

nodeId.egofeat - Binary features of the ego node.

nodeId.featnames - Anonymized names of the features (e.g., "anonymized feature 1").

3.2 Constructing the Similarity Graph

The first step in applying spectral clustering is constructing a similarity graph from the dataset. In the context of the Facebook Social Circles Dataset, the similarity graph represents the social network, where nodes represent Facebook users and edges represent friendships between users, as given in the dataset. Each edge indicates a connection, with a weight reflecting the similarity or strength of the relationship.



Read the Edge List - Extract the friendship connections from the nodeId.edges files or the combined edge file (facebook_combined.txt.gz). Construct an adjacency matrix A, where A_{ij}=1 if there is an edge between nodes i and j, and 0 otherwise.

Define Similarity Weights - For unweighted graphs, use the edge list directly to build the adjacency matrix. For weighted graphs, compute similarities using features from nodeId.feat or nodeId.egofeat.

Choose the Graph Type

- Fully Connected Graph - Connect all nodes with a similarity measure, suitable for smaller datasets but computationally intensive.
- k-Nearest Neighbors Graph (k-NN): Connect each node to its k-most similar neighbors, making the graph sparse and computationally efficient.
- Threshold Graph: Connect nodes only if their similarity exceeds a defined threshold.

Normalize the Graph (optional) - Normalize the adjacency matrix to balance the influence of nodes with varying degrees when constructing the Laplacian matrix.

3.3 Visualizing the Graph

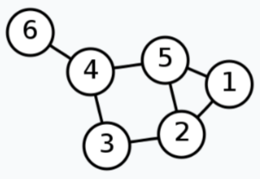
Visualizing the graph provides insights into the structure of the social network and helps interpret the relationships within the data. For the Facebook Social Circles Dataset, graph visualization can reveal densely connected communities, isolated nodes, or the overall connectivity of the network.

Use **NetworkX** or **igraph** in Python to construct and visualize the graph from the edge list. Apply a graph layout algorithm (**spring layout**, **Kamada-Kawai layout**, **Fruchterman-Reingold layout**) to position nodes in a way that highlights clusters and connectivity. Use node size, color, or labels to represent features. Color nodes based on predefined circles to compare with the clusters obtained through spectral clustering. For better readability, sample a subset of the graph or visualize a sparse version using a k-NN or thresholded graph.

4. Applying Spectral Clustering

4.1 Calculating the Laplacian Matrix

The **Laplacian matrix** is a critical component in spectral clustering, capturing the structural properties of the graph and enabling partitioning into clusters. It is derived from the graph's adjacency matrix A and degree matrix D.

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&3&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&1 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0&1&0\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&1\\ 1&1&0&1&0&0\\ 0&0&0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 2&-1&0&0&-1&0\\ -1&3&-1&0&-1&0\\ 0&-1&2&-1&0&0\\ 0&0&-1&3&-1&-1\\ -1&-1&0&-1&3&0\\ 0&0&0&-1&0&1 \end{pmatrix}$

Construct the **degree matrix** D, a diagonal matrix where $D_{ii}=\sum_j A_{ij}$ (the sum of edge weights for node i).

Unnormalized Laplacian ($L=D-A$) This is simple and effective for smaller, well-connected graphs.

Normalized Laplacians

- **Symmetric Normalized ($L_{sym}=I-D^{-1/2}AD^{-1/2}$)**, useful for balanced graphs.
- **Random Walk Normalized ($L_{rw}=I-D^{-1}A$)**, used for random walk interpretations of clustering.

For large, sparse graphs like the Facebook dataset, the normalized Laplacians are preferred for better numerical stability and interpretability. Normalize the graph if node degrees vary significantly to avoid bias from highly connected nodes.

4.2 Computing Eigenvectors

Eigenvectors of the Laplacian matrix play a central role in spectral clustering, as they provide a low-dimensional representation of the graph's structure that highlights potential clusters.

Compute the eigenvalues and eigenvectors of the Laplacian matrix L. Identify the eigenvectors corresponding to the k smallest non-zero eigenvalues. These eigenvectors represent the optimal low-dimensional embedding for clustering. The number of clusters k can be estimated using techniques like the eigengap heuristic (identifying a significant gap between consecutive eigenvalues). Form a feature matrix U, where each row corresponds to a node, and the columns are the selected eigenvectors. This matrix serves as the input for the clustering step.

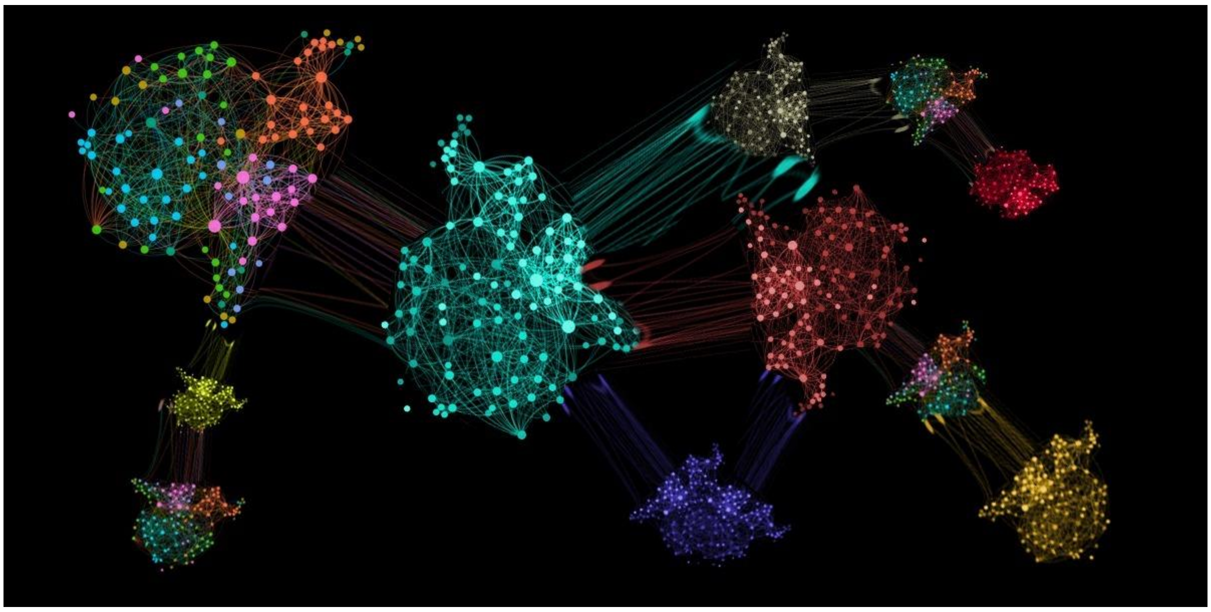
4.3 Implementing Clustering (k-means)

After computing the eigenvectors of the Laplacian matrix, the next step is to group the nodes into clusters using a traditional clustering algorithm like k-means. Use the feature matrix U, constructed from the k smallest eigenvectors of the Laplacian matrix, as the input for clustering. Each row of U represents a node in the graph in the reduced k-dimensional space. Use a k-means clustering algorithm to partition the nodes into k clusters. Initialize the algorithm with k, the number of clusters, and let it assign nodes to clusters by minimizing intra-cluster variance. The output of k-means is a set of cluster labels. Each node in the graph is assigned to one of the k clusters. Map the cluster labels back to the original graph to analyze the community structure of the network.

5. Evaluating Clustering Results

5.1 Visualizing Clusters

Visualizing the clusters is essential to understanding the results of spectral clustering and interpreting the community structures within the graph. Assign each node a color based on its cluster label obtained from the k-means algorithm. Use graph visualization tools like NetworkX, Matplotlib, or Plotly to plot the graph. Apply a layout to position nodes for better visibility of clusters.



Color nodes based on their cluster labels to distinguish different groups. Optionally, adjust node size or edge transparency to emphasize the clustering. Include a legend as well to indicate cluster labels. Try to annotate key nodes or clusters to provide additional information about the network.

5.2 Metrics for Cluster Evaluation

Evaluating the quality of clusters is critical to understanding how well the spectral clustering algorithm has partitioned the graph.

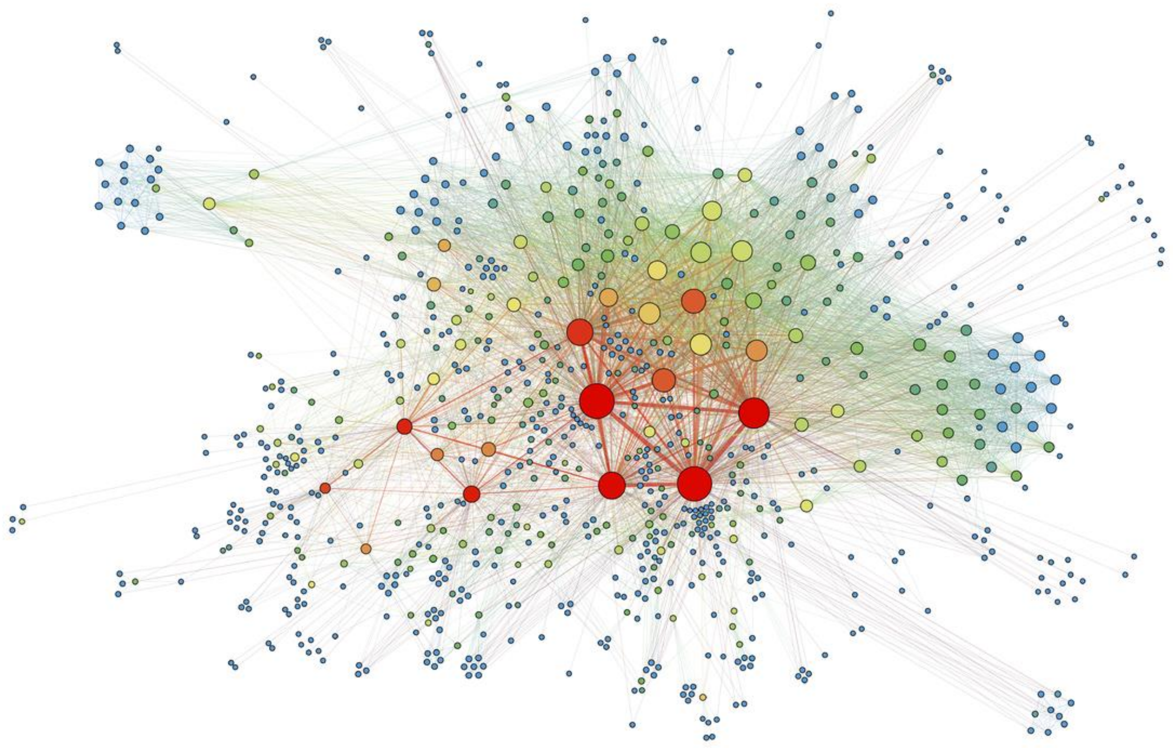
- **Silhouette Score**
  Measures how similar a node is to its own cluster compared to other clusters.
  Ranges from −1 to 1, where higher values indicate better-defined clusters.
- **Modularity**
  Quantifies the strength of division of a network into clusters by comparing the density of edges within clusters to edges between clusters.

Higher modularity values indicate stronger community structures.

- **Normalized Mutual Information (NMI)**
  Compares the clustering results to ground truth labels (e.g., social circles).
  Ranges from 0 to 1, where 1 indicates perfect agreement with the true labels.
- **Adjusted Rand Index (ARI)**
  Measures the similarity between predicted clusters and ground truth, accounting for chance.
  Ranges from −1 to 1, with higher values indicating better alignment with the true labels.
- **Intra-Cluster vs. Inter-Cluster Distance**
  Evaluate compactness (intra-cluster distance) and separation (inter-cluster distance).
  Smaller intra-cluster distances and larger inter-cluster distances indicate better clustering.

5.3 Interpreting the Results

Interpreting the results of spectral clustering involves analyzing the identified clusters to gain information regarding the structure and relationships within the network. Examine the nodes in each cluster to identify common patterns or shared attributes (e.g., similar features or connections). Compare clusters with the ground truth labels, such as predefined social circles, to validate the clustering. Assess the size and density of each cluster to understand their significance in the graph. Identify clusters representing tightly-knit communities versus loosely connected groups. Highlight key nodes (e.g., high-degree nodes or hubs) within clusters to understand their influence in the network.



Explore inter-cluster connections to identify bridging nodes that link different communities. Map the clusters to meaningful real-world interpretations, such as friend groups, interest-based communities, or other social structures.

6. Submission Guidelines

Prepare a comprehensive report that includes the following:

- Submit a well-documented **Python notebook** with comments explaining each step of your analysis. Use markdown cells to provide context and explanations.
- Summarize your findings, model, details of the clustering process, and conclusions inside the notebook. Focus on key points, challenges, results, and discussion.
- Any challenges you encountered during the entire process and how you overcame them.

Deliverables

- Detailed code along with markdown explanations, visualizations, results, discussion, key findings, and challenges. **(Jupyter / Python Notebook) (code.ipynb)**

Before submitting, compress the deliverables into a zip file. **(StudentName_012345.zip)**

Make sure to follow the naming convention for all deliverables as well as the final zipped file.

Note: All work submitted must be your own. Adherence to academic integrity is mandatory.

7. Grading Rubric

- Dataset Exploration and Preprocessing - **10 marks**
- Graph Construction - **25 marks**
- Spectral Clustering - **30 marks**
- Evaluating Clustering Results - **15 marks**
- Presentation of Results, Discussion, Explanations - **20 marks**