MSc. Data Science

Spectral Clustering based on Graph Laplacian

Authors:
Aalekhya Mukhopadhyay
MDS202301
Aishvarya S
MDS202302
Alena Maria Thomas
MDS202303
Aniruddha Bhattacharjee
MMA202208

Supervisor:
Prof. Priyavrat Deshpande
Lecturer
Chennai Mathematical Institute
pdespande@cmi.ac.in

Project Report

Contribution Table

Aalekhya Mukhopadhyay	Read and understood the paper. Understood and explained the paper from section 1 to 4. Implemented the algorithms and noted observations
Aishvarya S	Read and understood the paper. Understood and explained the paper of section 5 and implemented spectral clustering in section 8.
Alena Maria Thomas	Read and understood the paper. Understood and explained the paper of section 6.
Aniruddha Bhattacharjee	Read and understood the paper. Understood and explained the paper of section 7. Studied 3 texts and 1 paper for section 7 (cited in references).

Abstract

Spectral clustering is a powerful technique for identifying clusters within data by analyzing the eigenstructure of a similarity matrix. Unlike traditional clustering algorithms that rely on distance metrics in the original feature space, spectral clustering uses the eigenvectors of the similarity matrix to embed the data into a lower-dimensional space where clusters become more apparent.

1 Introduction

Spectral clustering is an unsupervised learning technique that leverages the spectral properties of a similarity graph to partition data into clusters. At the core of spectral clustering is the graph Laplacian, a matrix representation of the pairwise relationships between data points. The Laplacian matrix, denoted L, is defined as $L = D - A$, where A is the similarity matrix and D is the diagonal degree matrix containing the row sums of A. The Laplacian has several important properties that make it central to spectral clustering: 1. The eigenvalues of the Laplacian provide insights into the cluster structure of the data. The multiplicity of the eigenvalue 0 corresponds to the number of connected components in the similarity graph. 2. The eigenvectors of the Laplacian, can be used to embed the data into a lower-dimensional subspace where clusters are more easily separated. 3. The Laplacian is a positive semi-definite matrix, meaning its eigenvalues are non-negative real numbers. This makes the Laplacian amenable to spectral analysis.

2 Similarity Graphs and Graph Notation

2.1 The ϵ-neighbourhood graph

The ϵ-neighborhood graph is a method used in spectral clustering and other graph-based machine learning techniques to capture the local neighborhood structure of data points. In this approach, nodes in the graph represent individual data points, and an edge is formed between two nodes if the distance between their corresponding data points is less than a specified threshold value ϵ. One key characteristic of the ϵ-neighborhood graph is that it is typically treated as an unweighted graph. By representing the graph as unweighted, the ϵ-neighborhood graph focuses solely on encoding the local proximity structure, which is often the primary concern in spectral clustering and other graph-based algorithms that aim to identify clusters or communities based on neighborhood connectivity.

2.2 The k-nearest neighbor graphs

The k-nearest neighbor graph is another popular graph construction technique used in spectral clustering and related methods. Unlike the ϵ-neighborhood graph, which connects nodes based on a fixed distance threshold, the k-nearest neighbor graph aims to capture local neighborhood relationships by connecting each node to its k-nearest neighbors in the data space. However, the definition of k-nearest neighbors is inherently asymmetric, as node A may be among the k-nearest neighbors of node B, but B may not be among the k-nearest neighbors of A. This asymmetry leads to a directed graph, which is not desirable for spectral clustering algorithms that typically operate on undirected graphs. To overcome this issue, there are two common approaches to constructing an undirected k-nearest neighbor graph. The first approach is to simply ignore the edge directions and connect nodes A and B if either A is among the k-nearest neighbors of B, or B is among the k-nearest neighbors of A. The resulting graph is called the *k-nearest neighbor graph.* The second approach is more stringent and connects nodes A and B only if both A is among the k-nearest neighbors of B and B is among the k-nearest neighbors of A. This mutually inclusive condition results in the *mutual k-nearest neighbor graph.* In both cases, once the appropriate node pairs are connected, the edges are typically weighted by a similarity measure between the corresponding data points.

2.3 Fully connected graph

The fully connected graph is another approach to constructing graphs for spectral clustering, where all pairs of data points with non-zero similarity are connected by weighted edges. Unlike the ϵ-neighborhood graph or the k-nearest neighbor graph, which aim to capture local neighborhood relationships based on distance or proximity, the fully connected graph relies on the similarity function itself to model local neighborhoods. In this construction, an edge is formed between every pair of nodes (data points) with a positive similarity value, and the edge weight is set to the corresponding similarity value. This approach can be useful when the similarity function itself is designed to capture local neighborhood relationships within the data. One common example of such a similarity function is the Gaussian similarity function, defined as $s(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$, where σ is a parameter that controls the width or scale of the neighborhoods. Data points that are closer together in the feature space will have a higher similarity value, effectively modeling local neighborhoods centered around each data point. The parameter σ plays a similar role to the ϵ parameter in the ϵ-neighborhood graph, controlling the scale at which neighborhoods are defined. By adjusting σ, the Gaussian similarity function can capture different locality levels, from tight neighborhoods with small σ values to broader neighborhoods with larger σ values.

2.4 Graph Notation

An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set E. We will assume that the graph G is weighted, meaning each edge between vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$. The weighted adjacency matrix W represents these weights. For $v_i \in V$, the degree d_i of v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

The *degree matrix* D is a diagonal matrix with vertex degrees on the diagonal. For a subset of vertices $A \subset V$, the indicator vector is $1_A = (f_1, ..., f_n) \in \mathbb{R}^n$ where $f_i = 1$ if $v_i \in \mathbb{A}$ and $f_i = 0$ otherwise. $W(A, B)$ for two not necessarily disjoint sets $A, B \subset V$ is defined as,

$$W(A, B) := \sum_{i \in A, j \in B} w_{ij}$$

A subset A is connected if any two vertices in A can be joined by a path such that all intermediate points are also in A. A connected component is a connected subset with no connections to vertices outside the subset.

There are two ways of measuring the size of a subset $A \subset V$,

$$|A| := \text{number of vertices in } A$$

$$\text{vol}(A) := \sum_{i \in A} d_i$$

3 Graph Laplacians

The graph Laplacian matrices are the central tools employed in the spectral clustering technique. These matrices have garnered significant attention, leading to the emergence of a dedicated field known as spectral graph theory. We will focus on defining various variants of graph Laplacians and highlighting their most important properties. Here we will discuss unnormalized and normalized graph Laplacians.

3.1 Unnormalized Graph Laplacian

The unnormalized graph Laplacian matrix is defined as

$$L = D - W$$

where D is the degree matrix which is a diagonal matrix where d_{ii} is the degree of vertex i, and W is the adjacency matrix of the graph which represents the connections between the vertices, $w_{ij} = w_{ji} = 1$ if there is an edge between vertices otherwise $w_{ij} = w_{ji} = 0$.

(Proposition 1) Properties of L: *The matrix L satisfies the following properties:*

1. *For a vector $f \in \mathbb{R}^n$,*
$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$
2. *L is symmetric and positive semi-definite.*
3. *0 is the smallest eigenvalue of L, the constant one vector being the corresponding eigenvector.*
4. *L has n non-negative, real-valued eigenvalues.*

Proof:

1. For a vector f $\in \mathbb{R}^n$,
$$\begin{aligned} f'Lf &= f'(D-W)f = f'Df - f'Wf \\ &= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2}(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2) \\ &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \end{aligned}$$
2. W and D are symmetric matrices which implies L is also symmetric. From the proof in part 1, we see that $f'Lf \geq 0$ for all $f \in \mathbb{R}^n$, this proves L is positive semi-definite.
3. Parts 3 and 4 can be shown as consequences of the proofs in part 1 and 2 above.

(Proposition 2) Number of connected components and spectrum of L: *Given G is an undirected graph with non-negative weights, the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, ..., A_k$ in the graph. For eigenvalue 0, the eigenspace is spanned by the indicator vectors of the components denoted by $\mathbf{1}_{A_1}, ..., \mathbf{1}_{A_k}$.*

Proof: L has n eigenvalues $\lambda_1, ..., \lambda_n$ where n is the number of vertices in G. Suppose G has k connected components $A_1, ..., A_k$. The nullity of L is k which implies the multiplicity of the eigenvalue 0 is k. Each indicator vector $\mathbf{1}_{A_i}$ corresponding to the connected component A_i is an eigenvector of L with eigenvalue 0. Therefore, the eigenspace corresponding to the eigenvalue 0 is spanned by $\mathbf{1}_{A_1}, ..., \mathbf{1}_{A_k}$.

3.2 Normalized Graph Laplacian

There are two matrices which are called normalized graph Laplacians. One is a symmetric matrix denoted by L_{sym}, the other is closely related to a random walk and denoted by L_{rw}.

$$L_{sym} := D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$$
$$L_{rw} := D^{-1}L = I - D^{-1}W$$

(Proposition 3) Properties of L_{sym} and L_{rw}: *The normalized graph Laplacians satisfy the following properties:*

1. *For every $f \in \mathbb{R}^n$,*
$$f'L_{sym}f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}})^2$$

2. *λ is an eigenvalue of L_{rw} with eigenvector u $\iff$ λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}u$.*

3. *λ is an eigenvalue of L_{rw} with eigenvector u $\iff$ λ and u solve $Lu = \lambda Du$.*

4. *0 is an eigenvalue of L_{rw} with $\mathbf{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbf{1}$.*

5. *L_{sym} and L_{rw} have n non-negative real-valued eigenvalues and are positive semi-definite matrices.*

Proof:

1. For f $\in \mathbb{R}^n$,
$$\begin{aligned} f'L_{sym}f &= f'(I - D^{-1/2}WD^{-1/2})f \\ &= f'If - f'D^{-1/2}WD^{-1/2}f \\ &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2 \end{aligned}$$

2. When λ is an eigenvalue of L_{rw}, $L_{rw}u = \lambda u$. Then
$$\begin{aligned} L_{sym}w &= (I - D^{-1/2}WD^{-1/2})u \\ &= D^{1/2}u - D^{1/2}D^{-1}WD^{-1/2}D^{1/2}u \\ &= D^{1/2}u - D^{1/2}WD^{1/2}u \\ &= D^{1/2}(I - W)u \end{aligned}$$

This implies $w = D^{1/2}u$ is the eigenvector of $_{sym}$ with eigenvalue λ. The converse can be proved similarly.

3. Parts 3, 4 and 5 can be implemented as consequences of the proof in parts 1 and 2.

(Proposition 4) Number of connected components and spectra of L_{sym} and L_{rw}: *Given G is an undirected graph with non-negative weights, the multiplicity k of the eigenvalue 0 for both L_{sym} and L_{rw} equals the number of connected components $A_1, ..., A_k$ in the graph. For eigenvalue 0, the eigenspace of L_{rw} is spanned by the indicator vectors $\mathbf{1}_{A_1}, ..., \mathbf{1}_{A_k}$, and the eigenspace of L_{sym} is spanned by the vectors $D^{1/2}\mathbf{1}_{A_1}, ..., D^{1/2}\mathbf{1}_{A_k}$.*

Proof: L_{sym} and L_{rw} each have n eigenvalues $\lambda_1, ..., \lambda_n$ where n is the number of vertices in G. Suppose G has k connected components $A_1, ..., A_k$. For both L_{sym} and L_{rw}, nullity is k, which implies multiplicity of the eigenvalue 0 is k for both the Laplacians. Each indicator vector $\mathbf{1}_{A_i}$ corresponding to the connected component A_i is an eigenvector of L_{rw} with eigenvalue 0. Therefore, the eigenspace corresponding to the eigenvalue 0 is spanned by $\mathbf{1}_{A_1}, ..., \mathbf{1}_{A_k}$ for L_{rw}. From this we can conclude that the eigenspace of L_{sym} is spanned by $D^{1/2}\mathbf{1}_{A_1}, ..., D^{1/2}\mathbf{1}_{A_k}$.

4 Algorithms

We will study some spectral clustering algorithms, assuming our data consists of n arbitrary objects $x_1, ..., x_n$, and measuring the pairwise similarities $s_{ij} = s(x_i, xj)$ by a symmetric and non-negative similarity function and denoting the corresponding similarity matrix by $S = (s_{ij})_{i,j=1,...,n}$.

4.1 Unnormalized Spectral Clustering Algorithm

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$ and number of clusters k to be constructed.

1. Construct a similarity graph and let W be its weighted adjacency matrix.
2. Compute the unnormalized Laplacian matrix L.
3. Compute the first k eigenvectors $u_1, ..., u_k$ of L, and let U be the matrix containing the vectors $u_1, ..., u_k$ as columns.
4. Let $y_1, ..., y_n$ be the vectors corresponding to the n rows of U.
5. Cluster the points $y_i \in \mathbb{R}^k$, for $i = 1, ..., n$ into clusters $C_1, ..., C_k$.

Output: Clusters $A_1, ..., A_k$ with $A_i = \{j | y_j \in \mathbb{C}_i\}$.

4.2 Normalized Spectral Clustering Algorithm

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$ and number of clusters k to be constructed.

1. Construct a similarity graph and let W be its weighted adjacency matrix.
2. Compute the unnormalized Laplacian matrix L_{sym}.
3. Compute the first k eigenvectors $u_1, ..., u_k$ of L, and let U be the matrix containing the vectors $u_1, ..., u_k$ as columns.
4. Let us normalize the rows of U to norm 1, and form the matrix T which is set by $t_{ij} = u_{ij}/(\sum_k u_{ik}^2)^{1/2}$.

5. Let $y_1, ..., y_n$ be the vectors corresponding to the n rows of T.
6. Cluster the points $y_i \in \mathbb{R}^k$, for $i = 1, ..., n$ into clusters $C_1, ..., C_k$.

Output: Clusters $A_1, ..., A_k$ with $A_i = \{j | y_j \in \mathbb{C}_i\}$.

4.3 Observations:

- Deciding which similarity graph to choose from the fully connected graph, the k-nearest neighbor and the ϵ-neighborhood graph is a challenge. In the ϵ-neighborhood graph, it is difficult to choose an useful parameter ϵ. However, the k-nearest neighbor graph can connect points on different scales. On the other hand, the fully connected graph is very often used in connection with the Gaussian similarity function.
- Choosing the number of clusters k is a problem for all clustering algorithms. The choice of the number of clusters and the choice of the connectivity parameters of the neighborhood graph affect each other. Suppose, if the connectivity parameter is so small that the graph breaks into k connected components the choosing k as the number of clusters is a valid idea.

5 Spectral Clustering as an Approximation to Graph Partitioning Problems

Objective: To investigate the derivation of spectral clustering as an approximate solution to graph partitioning problems.

Spectral clustering partitions a similarity graph into groups such that points within the same group are similar and points in different groups are dissimilar. This can be achieved by minimizing the weight of edges between different groups and maximizing the weight of edges within the same group.

Considering we want to divide the graph into k groups, the mincut approach involves finding a specific division $(A_1, ..., A_k)$ which minimizes

$$cut(A_1, \dots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A, \overline{A})^2 \tag{1}$$

where $\overline{A}$ denotes the complement of A.

The mincut problem can sometimes lead to partitions where individual vertices are isolated from the rest of the graph, which is not desirable for clustering purposes. To address this, two common objective functions are used: RatioCut and normalized cut

$$RatioCut(A_1, \dots, A_k) := \frac{1}{k} \sum_{i=1}^{k} \frac{cut(A_i, \overline{A}_i)}{|A_i|} \tag{2}$$

$$Ncut(A_1, \dots, A_k) := \frac{1}{k} \sum_{i=1}^{k} \frac{cut(A_i, \overline{A}_i)}{vol(A_i)} \tag{3}$$

5.1 Ratiocut

Ratiocut minimizes the ratio of the total weight of edges between different subsets to the total number of vertices in the subsets.It does not incorporate normalization based on the size of the subsets and tends to produce partitions with roughly equal numbers of vertices in each subset. Also, sensitive to imbalance in the sizes of the subsets, potentially leading to unequal-sized clusters.

5.1.1 Approximating Ratiocut for k = 2:

Let $A \subset V$
We define the vector $f = (f_1, ...f_n)' \in \mathbb{R}^n$

$$fi = \begin{cases} \sqrt{\frac{|\overline{A}|}{|A|}} & v_i \in A \\ \text{-} \sqrt{\frac{|A|}{|\overline{A}|}} & v_i \in \overline{A} \end{cases}$$

Ratiocut objective function can be reformulated as

$$RatioCut(A, \overline{A}) = f'Lf \tag{4}$$

$$RatioCut(A, \quad \overline{A})|V| = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \tag{5}$$

Minimizing ratiocut is equivalent to minimizing $f'Lf$ subject to $f \perp \mathbf{1}$ and
$\parallel f \parallel = \sqrt{n}$

The optimization is still NP hard. The relaxation that can be done here is to remove discreteness condition and allow that f_i takes arbitrary values in $\mathbb{R}$.

RatioCut Goal: Divide a graph into balanced groups such that connections within groups are strong and connections between groups are weak.
Relaxation: The problem is mathematically reformulated using the Laplacian matrix (L) of the graph. This simplifies calculations.
Rayleigh-Ritz Theorem: This theorem helps find an approximate solution by analyzing eigenvalues and eigenvectors of L.
Key Result: The second smallest eigenvector of L (ignoring the smallest one which is always 0) corresponds to an approximate solution for RatioCut.

The previous approach of using the second eigenvector for RatioCut approximation becomes insufficient when dealing with more than two clusters ($k \geq 2$).

Instead,

Project onto Lower Dimension: The second eigenvector (or sometimes multiple eigenvectors) provides information about the data distribution. Treat as Data Points: These eigenvectors are interpreted as coordinates for each data point in a lower dimensional space. Apply k-means Clustering: The standard k-means clustering algorithm is used on these projected data points to assign them into k desired groups (C1, C2, ..., Ck).

5.1.2 Approximating RatioCut for arbitrary k:

Given a partition of V into k sets $(A_1, ..., A_k)$, we define k indicator vectors $h_j = (h_1, j, ...h_n, j)'$

$$f_{i,j} = \begin{cases} \frac{1}{\sqrt{|A_j|}} & , v_i \in A \\ 0 & , otherwise \end{cases}$$

H is n x k matrix and columns of H are orthonormal to each other, that is $H'H = I$

$$h\,'_i L h_i = \frac{Cut(A_i, A_i'}{|A_i|} = (H'LH)_{ii} \tag{6}$$

$$RatioCut(A_1, ..., A_k) = \sum h_i' L h_i = Tr(H'LH) \tag{7}$$

In spectral clustering, we often relax the problem by allowing the entries of a matrix H to take arbitrary real values. The relaxed problem aims to minimize the trace of $H'LH$, subject to the constraint $H'LH = I$, where L is a Laplacian matrix derived from the graph.

The solution to this relaxed problem is obtained by choosing H as the matrix containing the first k eigenvectors of L as columns. This is a standard form of a trace minimization problem, and it's analogous to the Rayleigh-Ritz theorem.

The matrix H obtained in this relaxed problem is equivalent to the matrix U used in the unnormalized spectral clustering algorithm. To obtain a discrete partition from the real-valued solution matrix H, the common approach is to apply the k-means algorithm on the rows of U.

5.2 N-cut

N-cut minimizes the ratio of the total weight of edges between different subsets to the total weight of edges within the subsets.It incorporates normalization based on the size of the subsets, aiming to balance between the volume of connections within subsets and the volume of connections between subsets and tends to produce partitions with subsets that are well-connected internally and weakly connected to other subsets. It is less sensitive to imbalance, as it takes into account the weights of edges within subsets.
The cluster indicator vector f is

$$fi = \begin{cases} \sqrt{\frac{vol(\overline{A})}{vol(A)}} & , v_i \in A \\ \text{-} \sqrt{\frac{vol(A)}{vol(\overline{A})}} & , v_i \in \overline{A} \end{cases}$$

Similar to above, $(Df)'\mathbf{1} = 0$
So
$min_{f \in R^n} f'Lf$ subject to $Df \perp \mathbf{1}, f'Df = vol(V)$.

For $k \geq 2$, we define the indicator vectors $h_j = (h_{(1,j)}, ...h(n,j)), where (i = 1, ..., n; j = 1, ..., k)$.
Therefore we minimize over all A_i $Tr(H'LH)$ subject to $H'DH = I$
Relaxing the discreteness condition, and taking T = $D^{1/2}H$, we minimize over all T

$Tr(T'D^{1/2}LD^{1/2}T)$ subject to $T'T = I$

6 Random walks point of view

A random walk on a graph is a stochastic process that jumps from vertex to vertex based on certain probabilities.

Here are some definitions that will be used in this section.

- The transition probability of jumping from vertex v_i to vertex v_j in one step is proportional to the edge weight w_{ij} and is given by

$$p_{ij} := \frac{w_{ij}}{d_i}$$

- The transition matrix $P = (p_{ij})_{i,j=1,\dots,n}$ of the random walk is thus defined by

$$P = D^{-1}W$$

- If the graph is connected and non-bipartite, then the random walk always possess a unique stationary distribution

$$\pi = (\pi_1, ..., \pi_n)', \text{where } \pi_i = \frac{d_i}{\text{vol}(V)}$$

- The transition matrix P is closely related to the normalized graph Laplacian $L_{rw} = I - P$, where I is the identity matrix.
  - Consequently, λ is an eigenvalue of L_{rw} with eigenvector $u \iff 1-\lambda$ is an eigenvalue of P with eigenvector u.
  - This also implies that the largest eigenvectors of P and the smallest eigenvectors of L_{rw} can be used to describe the cluster properties of the graph.

6.1 Random walks and Ncut

This section explores how spectral clustering can be interpreted as trying to find a partition of the graph such that the random walk stays long within the same cluster and seldom jumps between clusters.

(Proposition 5) Ncut **via transition probabilities** *Let G be connected and non bi-partite. Assume that we run the random walk $(X_t)_{t\in\mathbb{N}}$ starting with X_0 in the stationary distribution π. For disjoint subsets $A, B \in V$, denote by $P(B|A) := P(X_1 \in B|X_0 \in A)$. Then:*

$$\text{Ncut}(A, \overline{A}) = P(\overline{A}|A) + P(A|\overline{A})$$

Proof.

$$\begin{aligned}
P(X_0 \in A, X_1 \in B) &= \sum_{i\in A, j\in B} P(X_0 = i, X_1 = j) \\
&= \sum_{i\in A, j\in B} \pi_i p_{ij} \\
&= \sum_{i\in A, j\in B} \frac{d_i}{\text{vol}(V)} \frac{w_{ij}}{d_i} \\
&= \frac{1}{\text{vol}(V)} \sum_{i\in A, j\in B} w_{ij}
\end{aligned}$$

Using this, we obtain

$$\begin{aligned} P(X_1 \in B | X_0 \in A) &= \frac{P(X_0 \in A, X_1 \in B)}{P(X_0 \in A)} \\ &= \left(\frac{1}{\text{vol}(V)} \sum_{i \in A, j \in B} w_{ij} \right) \left(\frac{\text{vol}(A)}{\text{vol}(V)} \right)^{-1} \\ &= \frac{\sum_{i \in A, j \in B} w_{ij}}{\text{vol}(A)} \end{aligned}$$

Hence, this proposition proves the existence of a formal equivalence between the normalized cut (Ncut) criterion and the transition probabilities of random walks. The Ncut criterion aims to find a partition of the graph that minimized the cut between clusters while maximizing the connections within clusters. This proposition established that when minimizing Ncut, we actually look for a cut through the graph such that a random walk seldom transitions from A to $\overline{A}$ and vice versa.

6.2 Commute distance

The commute distance (also called resistance distance) between two vertices v_i and v_j is the expected time it takes the random walk to travel from vertex v_i to vertex v_j and back.

The commute distance has several nice properties which make it favourable for machine learning:

- As opposed to the shortest path distance on a graph, the commute distance between two vertices decreases if there are many different short ways to get from vertex v_i to vertex v_j. So, instead of just looking for the shortest path, the commute distance considers all possible paths between vertices, reflecting the graph's underlying structure.
- Commute distance is particularly appealing for clustering as it captures relationships between vertices within high-density regions of the graph. Points which are connected by a short path in the graph and lie in the same high-density region of the graph are considered closer to each other than points which are connected by a short path but lie in different high-density regions of the graph.

The commute distance of a graph can be computed with the help of the generalized inverse (also called pseudo-inverse or Moore-Penrose inverse) $L^\dagger$ of the graph Laplacian L. Recall that by Proposition 1, the matrix L can be decomposed as $L = U\Lambda U'$, where U is the matrix containing all eigenvectors as columns and Λ is the diagonal matrix with the eigenvalues $\lambda_1, ..., \lambda_n$ on the diagonal. As at least one of the eigenvalues is 0, the matrix L is not invertible. Instead, we define its generalized inverse as $L^\dagger := U\Lambda^\dagger U'$ where the matrix $\Lambda^\dagger$ is the diagonal matrix with diagonal entries $1/\lambda_i$ if $\lambda_i \neq 0$ and 0 if $\lambda_i = 0$. The entries of $L^\dagger$ can be computed as $l^\dagger_{ij} = \sum_{k=2}^{n} \frac{1}{\lambda_k} u_{ik} u_{jk}$. The matrix $L^\dagger$ is positive semi-definite and symmetric.

(Proposition 6) Commute distance *Let $G = (V, E)$ be a connected, undirected graph. Denote by c_{ij} the commute distance between vertex v_i and vertex v_j, and by $L^\dagger = (l^\dagger_{ij}) - i, j = 1, ..., n$ the generalized inverse of L. Then, we have:*

$$c_{ij} = \text{vol}(V)(l^\dagger_{ii} - 2l^\dagger_{ij} + l^\dagger_{jj}) = \text{vol}(V)(e_i - e_j)' L^\dagger (e_i n = e_j)$$

where $e_i = (0, ..., 0, 1, 0, ..., 0)'$ is the i-th unit vector.

The proof using first step analysis of random walks can be found in [3].

Proposition 6 establishes a direct relationship between commute distance and L†. It shows that $\sqrt{c_{ij}}$ can be taken to be a Euclidean distance function on the vertices of the graph. This approach enables the construction of an embedding that maps the vertices v_i of the graph on points $z_i \in \mathbb{R}^n$ such that the Euclidean distances between the points z_i align with the commute distance on the graph, thus facilitating clustering algorithms that rely on distance-based measures.

Here's how this embedding works: Since the matrix $L^\dagger$ is positive semi-definite and symmetric, it induces an inner product on $\mathbb{R}^n$, particularly on the subsapce orthogonal to the constant vector $\mathbf{1}$. By selecting z_i as the point corresponding to the i-th row of the matrix $U(\Lambda^\dagger)^{1/2}$. Then by Proposition 6 and definition of $L^\dagger$, we get that $< z_i, z_j >= e_i' L^\dagger e_j$ and thus $c_{ij} = \text{vol}(V)||z_i - z_j||^2$. This provides a geometric representation of the commute distances, facilitating analysis and visualization of the graph's structure in Euclidean space.

The embedding used in unnormalized spectral clustering is related to the commute time embedding, but not identical. In spectral clustering, the vertices of the graph are mapped on the rows y_i of the matrix U, while the commute time embedding maps the vertices on the rows z_i of the matrix $(\Lambda^\dagger)^{1/2}U$. That is, the entries of z_i are additionally scaled by the inverse eigenvalues of L. Moreover, spectral clustering only considers the first k columns of the matrix, whereas the commute time embedding utilized all columns. Some researchers argue that despite these differences, both approaches aim to construct clusters based on Euclidean distances between y_i or z_i, respectively, and interpret this as clustering vertices based on commute distance. However, substantial differences can arise. For instance, in the ideal scenario of k disconnected components, the first k eigenvalues of L are 0 (by Proposition 2) and the first k columns of U consist of the cluster indicator vectors. However, the first k columns of the matrix $(\Lambda^\dagger)^{1/2}U$ consist of zeros only, as the first k diagonal entries of $\Lambda^\dagger$ are 0. Thus, the information contained in the first k columns of U is completely ignored in the matrix $(\Lambda^\dagger)^{1/2}U$, and all the non-zero elements of the matrix $(\Lambda^\dagger)^{1/2}U$ which can be found in the columns $k+1$ to n are disregarded by spectral clustering. On the other hand, these discrepancies are absent in connected graphs. In such cases, where the only eigenvector with eigenvalue 0 is the constant vector, both approaches disregard it. The matrix $(\Lambda^\dagger)^{1/2}U$ emphasized eigenvalues corresponding to small eigenvalues of L, magnifying their influence. Hence, while similarities exist, discrepancies arise, especially in disconnected graphs.

7 Observations from perturbation theory

This section pertains to how eigenpairs of a matrix A change once a minor *perturbation* H is added to it. If this new *perturbed* matrix is denoted by $\widetilde{A} := A + H$, we claim for the time being that the new eigenpairs are *close enough* to the old ones, and then develop the rationale for our domain to analyse the efficacy of the k-means algorithm.

We first look at the *best-case scenario* where the dataset gives us exactly zero inter-cluster similarity. As seen earlier, the indicator vectors of these clusters are the first k eigenvectors of L

or L_{rw}, with the vectors $y_i \in \mathbb{R}^k$ constructed via the algorithm being

$$y_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The 1 is indexed i i.e. indicates the connected component it is sitting in. In particular, within each connected component, all y_i are the same, and the k-means algorithm trivially partitions the points by placing a center point on each of the distinct y_i.

Now for a situation where we still have distinct clusters with *small enough* inter-cluster similarity, we apply our perturbation approach to draw some observations. We treat our Laplacians $\widetilde{L}$ or $\widetilde{L_{rw}}$ here as having been perturbed from the L or L_{rw} i.e. $\widetilde{L} = L + H$ or $\widetilde{L_{rw}} = L_{rw} + H$ for some perturbation matrix H. Following from our claim at the beginning of the section, the new eigenpairs are *close enough* to the old eigenpairs, with the new y_i being equal to the old indicator vectors upto a *small enough* error term. Assuming the perturbation to be not too large, the clustering technique still succeeds.

We will now see what being *close enough* actually means. To give an intuitive idea, we are looking at a statement akin to "*distance* between eigenpairs is bounded by a constant multiple of the norm of H, the perturbation matrix", with the constant depending on the eigenvalue in consideration and its distance from the rest of the spectrum. The rigorous mathematical statement follows below.

7.1 The formal statement

Our claim above is rigorously treated under the remit of matrix perturbation theory, namely by the Davis-Kahan theorem. Before we state the result, we develop the notion of *distances between spaces*, which is measured using *canonical angles* or *principal angles.*

Given $\mathcal{V}_1, \mathcal{V}_2$, two p-dimensional subspaces of $\mathbb{R}^d$ with corresponding orthonormal matrices V_1, V_2, we define the cosines $\cos\Theta_i$ of the *principal angles* Θ_i as the singular values of $V_1^T V_2$. Note that if V_1, V_2 are 1-dimensional spaces, this notion falls in line with the usual definition of the angle between two planes. This definition can also be extended to cases where $\mathcal{V}_1, \mathcal{V}_2$ are of varying dimensions; however such is beyond the scope of the discussion here, and can be studied at [6, Section V], [1, Section VII.3] and [4, Section 12.4.3].

As for notation, $\sin\Theta(\mathcal{V}_1, \mathcal{V}_2) := diag(\sin\Theta_1, \cdots, \sin\Theta_i, \cdots)$

Theorem. *(Davis-Kahan) Let $A, H \in \mathbb{R}^{n\times n}$ be symmetric matrices, and let $\|\cdot\|$ be the Frobenius norm or the two-norm for matrices. Consider $\widetilde{A} := A + H$ as a perturbed version of A. Let $S_1 \subset \mathbb{R}$ be an interval. Define $\sigma_{S_1}(A) := (A) \cap S_1$, and V_1 as the eigenspace corresponding to the eigenvalues in $\sigma_{S_1}(A)$ (i.e. the image of the spectral projection induced by $\sigma_{S_1}(A)$. Denote by $\sigma_{S_1}(\widetilde{A}), \widetilde{V_1}$ the analogous quantities for $\widetilde{A}$. Define the distance between S_1 and $(A) \setminus S_1$ as*

$$\delta := \min\{|\lambda - s| : \lambda \in (A) \setminus S_1, s \in S_1\}.$$

Then the distance $d(V_1, \widetilde{V_1}) := \det(\sin\Theta(\mathcal{V}_1, \mathcal{V}_2))$ between the two subspaces V_1 and $\widetilde{V_1}$ is bounded by

$$d(V_1, \widetilde{V_1}) \leq \frac{\|H\|}{\delta}.$$

The proof can be found at [6, Section V.3], and the reader is encouraged to explore it as a detour. With the matter at hand, we look to apply this theorem to the case of the unnormalised Laplacian (with the normalised Laplacian case following suit).

The matrix A now refers to the graph Laplacian of the *best-case scenario*, where the graph has k connected components. Now the perturbed matrix $\widetilde{A}$ corresponds to the second case dealt with earlier, where the individual clusters are not completely isolated, but have only few edges with low weight; so $\widetilde{A} = \widetilde{L}$.

Now choosing the interval S_1 becomes an important step, such that the first k eigenvalues of both L and $\widetilde{L}$ lie within S_1. If such a choice is possible, we can invoke the above theorem immediately to get that the eigenspaces corresponding to the first k-eigenvalues of *best-case* L and *perturbed-case* $\widetilde{A}$ are close to each other. Moreover, the eigenvectors of L being piece-wise constant on the connected component tells us the behaviour is mimicked for $\widetilde{L}$ i.e. the eigenvectors of $\widetilde{L}$ are *approximately* constant on the connected components. The error here depends on broadly two factors: the perturbation norm $\|H\|$, and the distance δ between S_1 and the $(k+1)^{th}$ eigenvalue of L [pick $S_1 := [0, \lambda_k]$, where λ_k is the k^{th} eigenvalue, then $\delta = |\lambda_k - \lambda_{k+1}|$, the eigengap].

Note from the discussion above that if the eigengap is large or the perturbation norm is small, then the spectral clustering is efficient. However, in the opposite case, the choice of S_1 might be a hurdle. If the eigengap (or δ) ends up being too small, then the bound becomes too huge to be of any use. If this can be avoided, one way to bypass further issues is by choosing S_1 containing the first k-eigenvalues of L and the first $\widetilde{k}(\neq k)$ eigenvalues of $\widetilde{L}$. This is still a weaker case to work with. However this analysis leads us to believe that the eigengap can, in fact, be used as a quality criterion for the clustering, which is elaborated upon in the following segment.

7.2 Further comments on the perturbation approach

A natural alternative to our work so far is to work with the similarity or adjacency matrices instead of the graph Laplacian. Indeed, these are block-diagonal symmetric matrices, which in turn yield an eigenbasis which are real-valued inside the individual blocks and zero outside. While this is a necessary condition, sufficiency is ill-advised because of two more factors that can improve the clustering:

- Since we pick the *first* k eigenvalues, we need to make sense of the *order*. Comparatively speaking, the Laplacian ensures that each connected component yields exactly one eigenpair with eigenvalue 0, so enumerating the *first* k eigenvalues (for k connected components in the graph) gives us a unique eigenvector per component, as opposed to working with similarity or adjacency matrices, where two eigenvalues (say, the largest) come from the same block. Such a situation could lead to over-representation of some blocks and under-representation of others, when considering the *first* k eigenvalues.

- The *best-case* eigenvectors on the components should have elements *far away* from 0. Note that in the *best-case scenario*, a nonzero element u indexed (c, i) tells us that the i^{th} point belongs to the c^{th} cluster, and vice versa. Now, as the *perturbed case* is close to the *best case*, if $u \approx 0$, then it is difficult to say where $\widetilde{u}$ lands. If u is small enough to be considered zero but not $\widetilde{u}$, then the i^{th} point belongs to the c^{th} cluster in the *perturbed case* but not in the *best case*, and vice versa, leading to mis-clustering.

Note that for both L and L_{rw}, the relevant eigenvectors for the *best case* are indicator vectors, so the second problem is circumvented. However using L_{sym} can be tricky, as in the normalised spectral clustering algorithm of [5]. The eigenvectors in the *best case* are $\sqrt{D}\mathbb{1}_{A_i}$. Now if there are vertices having low degree, the corresponding entries in the eigenvectors become close to 0. This is dealt with by row-normalising in the algorithm, wherein the matrix having exactly one nonzero entry per row transforms to a matrix with indicator vectors. There is a subtle hindrance here, wherein if a row has nonzero entries ϵ_1, ϵ_2, then row-normalising introduces a factor of $\frac{1}{\sqrt{\epsilon_1^2+\epsilon_2^2}}$, which blows up the element, leading to a mis-clustering situation similar to above. However, observe that such a situation occurs in case of low-degree vertices (eigenvectors are given by $\sqrt{D}\mathbb{1}_{A_i}$), leading to such data points practically being outliers anyway.

In conclusion, unnormalised and normalised spectral clustering with L_{rw} sit well in line when analysed by perturbation theory. Normalised spectral clustering with L_{sym} also works practically well, with some care exercised in case of graphs with low-degree vertices.

8 Implementation of spectral clustering from scratch

Step 1: Compute pairwise distance matrix
Step 2: Compute similarity matrix
Step 3: Compute the Laplacian matrix
Step 4: Compute the eigenvectors and eigenvalues of the Laplacian matrix
Step 5: Select the first k eigenvectors corresponding to the smallest eigenvalues
Step 6: Normalize rows of the eigenvector matrix
Step 7: Perform KMeans clustering on the normalized eigenvector matrix

Link : https://colab.research.google.com/drive/1z73jwn5TzN_lyJfYctnj0ViG9Guabr80?usp=sharing

9 Conclusion

In conclusion, both normalized and unnormalized graph Laplacians serve as powerful tools in spectral clustering algorithms, offering distinct advantages and applications in various contexts. Unnormalized Laplacians, such as the classic Laplacian matrix, provide a direct representation of the graph's structure and are widely used in clustering tasks. By capturing the pairwise relationships between vertices and their neighborhoods, unnormalized Laplacians offer an intuitive approach to partitioning the graph. On the other hand, normalized Laplacians, including the random walk and symmetric normalized Laplacians, offer enhanced spectral properties that account for differences in node degrees. By scaling the edges or vertices based on their degrees, normalized Laplacians mitigate the effects of node degrees on the spectral decomposition, resulting in improved clustering performance, particularly in graphs with highly varying node degrees.

Spectral clustering offers an approximate solution to graph cut problems like RatioCut. It leverages the Laplacian matrix and eigenvalues to find a good separation between clusters. While the second eigenvector provides an initial approximation for two clusters, for more than two clusters ($k \geq 2$), spectral clustering projects data points into a lower-dimensional space based on eigenvectors and then uses k-means clustering to assign points to their final groups. This approach combines the strengths of spectral clustering for structure discovery with the efficiency of k-means for final cluster assignment.

Spectral clustering algorithms leverage the eigenvalues and eigenvectors of Laplacian matrices to partition the graph into cohesive clusters. By embedding the graph's structure into a low-dimensional spectral space, spectral clustering facilitates effective partitioning, even in the presence of complex connectivity patterns and overlapping clusters.

By leveraging the transition probabilities of random walks, spectral clustering effectively identifies cohesive clusters within the graph. Furthermore, the utilization of commute distance as a measure of similarity offers a novel perspective on graph clustering, enabling the construction of meaningful embeddings. Despite the intricacies involved, the integration of random walk theory enhances our ability to uncover and understand complex patterns within graph data, thereby advancing the field of machine learning and data mining.

We also look at spectral clustering through the lens of perturbation theory, pulling observations in situations close to optimal cases. We further appreciate why L and L_{rw} are frontrunners to be considered for this, followed by L_{sym}, while discouraging the use of similarity or adjacency matrices, all while building robust reasoning with progressive intuition.

To close it all out, we look at the implementing the theory step-by-step from ground zero, with the Colab project linked.

References

[1] R. Bhatia. *Matrix Analysis*. Graduate Texts in Mathematics. Springer New York, 1996.

[2] Changyao Chen. Spectral clustering, step by step. https://changyaochen.github.io/spectral-clustering/.

[3] François Fouss, Alain Pirotte, and Marco Saerens. The application of new concepts of dissimilarities between nodes of a graph to collaborative filtering. *ACM Transactions on Mathematical Software - TOMS*, 01 2004.

[4] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[5] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[6] G.W. Stewart and J. Sun. *Matrix Perturbation Theory*. Computer Science and Scientific Computing. Elsevier Science, 1990.

[7] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.