Politechnika Krakowska
im. Tadeusza Kościuszki
Wydział Informatyki i Telekomunikacji

Jakub Ptak

numer albumu: 143243

Wybrane algorytmy spektralnej analizy skupień: podstawy matematyczne, zastosowania, implementacje

Some spectral clustering algorithms: mathematical foundations, applications in practice, implementations

praca inżynierska

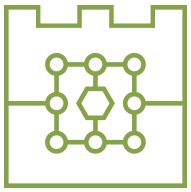
na kierunku Matematyka stosowana

specjalność analityka danych

Praca przygotowana pod kierunkiem:
dr Marcina Skrzyńskiego,
mgr Adriana Kowalskiego

Kraków, 2025

Cracow University of Technology
Faculty of Computer Science and Telecommunications



Jakub Ptak

album nr: 143243

Wybrane algorytmy spektralnej analizy skupień: podstawy matematyczne, zastosowania, implementacje

Some spectral clustering algorithms: mathematical foundations, applications in practice, implementations

engineering thesis

in the field of Applied Mathematics

specialization data analytics

Work prepared under the supervision of:
dr Marcin Skrzyński,
mgr Adrian Kowalski

Cracow, 2025

Contents

1. Introduction

Nowadays, the data size has grown to the scale where computing time and cost are significant. Without an optimized approach, multi-dimensional data can be a hindrance to analytics. A subsequent solution for given problem is the reduction of the dimension of data. Using spectral clustering algorithms we are able to reduce the number of columns used in the cluster assignment process to a necessary minimum. Our goal is to divide multi-dimensional points of data into groups with respect to their similarity. Those algorithms find use in most of the fields of data science, as the complexity of data sets has became an ordinarity, and therefore it does not surprise, it is a widely spoken topic among mathematicians. My thesis encompasses the theoretical explanation of the linear algebra, especially matrix-related theorems, and graph theory used for the spectral reduction algorithms. It takes insights into available and well-known algorithms that are used for such purpose. In the "Literature review" section, having provided necessary theory, we distinguish three main algorithms. Two of them are spectral clustering algorithms using graph Laplacian, normalized and unnormalized. The third algorithm discussed is the standard k-means algorithm used as a point of reference. The "Methodology" section shall provide a comprehensive comparison of previously mentioned algorithms using supervising learning methods.

2. Literature review

2.1. Fundamental definitions

In this chapter we describe fundamental definitions and introduce essential notations. The more complex concepts shall be described in detail in further subsections. Let Y be a set. The cardinality of the set Y is denoted as $|Y|$. Using $\mathbb{N}$, we represent the set of natural numbers. Using $\mathbb{R}$, we represent the set of real numbers. Using $\mathbb{C}$, we represent the set of complex numbers. We shall always denote natural logarithm as ln. Then the addition in V satisfies the following properties: for all $u, v, w \in V$ we have $(u+v)+w = u+(v+w)$ (associativity) and $u+v = v+u$ (commutativity), there exists a zero vector $\vec{0}$ such that for any $v \in V$ we have $v + \vec{0}$, for any $v \in V$ there exists an opposite vector $-v \in V$ such that $v + (-v) = \vec{0}$. Furthermore, the scalar multiplication in V satisfies $c \cdot (u+v) = c \cdot u + c \cdot v$, $(c+d) \cdot v = c \cdot v + d \cdot v$, $(c \cdot d) \cdot v = c \cdot (d \cdot v)$, and $1 \cdot v = v$, where $c, d \in \mathbb{F}$, $u, v \in V$, and 1 is the multiplicative identity in $\mathbb{F}$.

In a finite-dimensional space, $v \in \mathbb{F}^n$ can be written as:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}, \quad \text{where } v_i \in \mathbb{F}.$$

2.2. Graphs

Spectral clustering algorithms rely on graph theory, in this paper we will be using similarity graphs to elegantly represent divisions between groups of points. The main literature for this part is Luxbourg [3].

Definition 2.2.1. We define graph as a pair $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is a nonempty finite set called the set of vertices, and E is a set of unordered pairs of vertices, referred to as edges. Instead of "V is the set of vertices of G" we will write $V = V(G)$

Definition 2.2.2. A directed graph is a graph in which the elements of E are ordered pairs. For any $a, b \in V$, if $(a, b) \in E$, then it is not necessarily true that $(b, a) \in E$.

Definition 2.2.3. A simple graph is an undirected graph without multiple edges, where each edge connects two distinct vertices. A complete graph is a graph where any two distinct vertices are connected by an edge.

Definition 2.2.4. A subgraph of a graph $G = (V, E)$ is a graph obtained by removing vertices or edges from G. If a vertex $a \in V$ is removed, all edges containing a are also removed.

Definition 2.2.5. A clique is a subgraph of G that is a complete graph.

Definition 2.2.6. The adjacency matrix A of a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ is a binary square matrix $A = [a_{ij}]$ of size n defined by $a_{ij} = 1$ if $(v_i, v_j) \in E$, and $a_{ij} = 0$ otherwise. For weighted graphs we can also introduce weighted adjacency matrix W where if $(v_i, v_j) \in E$ instead of $a_{ij} = 1$, a_{ij} is equal to the weight of an edge from the vertex v_i to the vertex v_j.

Definition 2.2.7. The degree of a vertex v in a simple graph $G = (V, E)$ is defined as the number of all vertices $u \in V$ such that $(v, u) \in E$. (These vertices are referred to as adjacent to v). We will denote the degree of v by $d(v)$.

Definition 2.2.8. The degree matrix of a simple graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ is defined to be the diagonal matrix D of size n whose ith diagonal entry is equal to $d(v_i)$.

Definition 2.2.9. For any set of vertices $A \subset V$, its complement $V \setminus A$ is denoted by $\bar{A}$.

Definition 2.2.10. Let $G = (V, E)$ be a graph with $V = \{v_1, \ldots, v_s\}$. For a set $A \subset V$, we introduce the indicator vector $\mathbf{A} = (f_1, \ldots, f_n) \in \mathbb{R}^n$, where $f_i = 1$ if $v_i \in A$, and $f_i = 0$ otherwise. To simplify notation, we will write $i \in A$ to refer to the situation where $v_i \in A$, particularly when working with sums like $\sum_{i \in A} w_{ij}$.

Definition 2.2.11. Let $G = (V, E)$ be a weighted graph with the weighted adjacency matrix $W = [w_{ij}]$. For two (potentially overlapping) subsets $A, B \subset V$, we define:

$$W(A, B) := \sum_{i \in A} \sum_{j \in B} w_{ij}. \tag{2.1}$$

We will use two distinct methods to measure the "size" of a subset $A \subset V$. The first is $|A|$, the number of vertices in A. Or, $\mathrm{vol}(A) := \sum_{v \in A} d(v)$ the sum of the degrees of vertices in A.

In other words, $|A|$ measures the size of A by counting its vertices, while $\mathrm{vol}(A)$ measures it by counting the edges connected to vertices in A.

Definition 2.2.12. Let $k \in \mathbb{N}$.Then $B_k(w)$ represents the set of vertices of the graph G that are within a distance of at most k from a chosen vertex $w \in V(G)$

2.3. Different types of similarity graphs

Now, as we have discussed the necessary theory for graphs we need to bring information about them. The data we receive rarely comes in the form of a graph and therefore we need to obtain one using one of the available methods. As our goal in the clustering exercise is to group points $x_1, \ldots, x_n$ (typically in $\mathbb{R}^\kappa$)pairwise with respectively, either s_{ij} pairwise similarities or d_{ij} pairwise distances. Hence, this point of the paper encompasses some chosen, popular methods to do so [3].

1. The ε-neighbourhood graph - Nodes x_i and x_j are connected if their distance d_{ij} satisfies:

$$d_{ij} \leqslant \varepsilon,$$

where $\varepsilon > 0$ is a predefined threshold. The weight of the edge is often given by a similarity function such as:

$$w_{ij} = \begin{cases} \text{similarity}(x_i, x_j), & \text{if } d_{ij} \leqslant \varepsilon, \\ 0, & \text{otherwise.} \end{cases}$$

This method may result in disconnected graphs if *varepsilon* is too small.

2. k-Nearest neighbour graph - Each node x_i is connected to its k nearest neighbours based on a distance metric (e.g., Euclidean distance). There are two main variants:

— **Directed k-Nearest Neighbour Graph:** The connection is directed from node x_i to node x_j if x_j is among the k nearest neighbour of x_i.
— **Mutual k-Nearest Neighbour Graph:** An edge is added between x_i and $x_j$if only if x_i is among the k nearest neighbours of x_j and vice versa.

The weights of the edges can be defined using a similarity function, exempli gratia, a Gaussian kernel:

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right),$$

where $||\cdot||$ is Euclidean norm in $\mathbb{R}^n$.

3. The fully connected graph - In this method, all nodes are connected to each other, and the weights of the edges are determined by a similarity function:

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right).$$

This approach captures the global structure of the data but can be computationally expensive for large datasets.

2.4. Vector operations

The key elements of spectral clustering are vector operations, they would be further used to obtain information from previously defined graphs and matrices describing them.

Definition 2.4.1. The Euclidean *norm* of a vector $v \in \mathbb{R}^n$, denoted $\|v\|$, represents its length and is defined as:

$$\|v\| = \sqrt{v \cdot v},$$

where $v \cdot v$ is the dot product of v with itself.

Definition 2.4.2. The *dot product* (or inner product) of two vectors $v, w \in \mathbb{R}^n$ is defined as:

$$v \cdot w = \sum_{i=1}^{n} v_i w_i,$$

where v_i and w_i are the components of v and w, respectively. The dot product has the following properties: if $u, v, w \in \mathbb{R}^n$, then

1. $v \cdot w = w \cdot v$ (commutativity),
2. $(\alpha v) \cdot w = \alpha(v \cdot w)$ for $\alpha \in \mathbb{R}$ (mixed associativity),
3. $v \cdot (w + u) = v \cdot w + v \cdot u$ (distributivity over addition),
4. $v \cdot v \geqslant 0$, and $v \cdot v = 0 \iff v = 0$.

Definition 2.4.3. A set of vectors $\{v_1, v_2, \ldots, v_k\}$ in $\mathbb{R}^n$ is *linearly independent* if for any $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$ the following implication holds true

$$\sum_{i=1}^{k} \alpha_i v_i = 0 \implies \alpha_1 = \alpha_2 = \cdots = \alpha_k = 0,$$

The linear *span* of the set of vectors is the set of all linear combinations of those vectors:

$$\mathrm{Span}_{\mathbb{R}}(v_1, v_2, \ldots, v_k) = \left\{ \sum_{i=1}^{k} \alpha_i v_i \mid \alpha_i \in \mathbb{R} \right\}.$$

Definition 2.4.4. Two vectors $v, w \in \mathbb{R}^n$ are *orthogonal* if $v \cdot w = 0$. A set of vectors is *orthonormal* if they are pairwise orthogonal and each vector has unit length ($\|v\| = 1$). Instead of "vector v is orthogonal to vector w" it is accepted to state: $v \perp w$. Due to the commutativity of the inner product it is true that $\forall\ v, w \in \mathbb{R}^n : v \perp w \iff w \perp v$

Theorem 2.4.5. *If $v, w \in \mathbb{R}^n$ are not non-zero, the following statement is true:*

$$\measuredangle(v, w) = \frac{\pi}{2} \iff v \perp w. \tag{2.2}$$

Theorem 2.4.6 (Pythagorean theorem)**.** *Let $u, v, w \in \mathbb{R}^n$ and let $\alpha \in \mathbb{R}$, then:*

1. $\overrightarrow{0} \perp v$,
2. $v \perp v \iff v = \overrightarrow{0}$,
3. $v \perp w \implies \alpha v \perp w$,
4. $v \perp u \wedge v \perp w \implies v \perp (u \pm w)$.

Theorem 2.4.7. *Let $v, w \in \mathbb{R}^n$. Then the following conditions are equivalent:*

(I) $v \perp w$,
(II) $\|v + w\|^2 = ||v||^2 + ||w||^2$,
(III) $||v - w||^2 = ||v||^2 + ||w||^2$.

Proof. We shall first prove the equivalence (I) $\iff$ (II).

Starting with (II), the norm:

$$\text{(II)} \iff ||v + w||^2 = ||v||^2 + ||w||^2.$$

Using the definition of the squared norm, we have:

$$||v + w||^2 = (v + w) \cdot (v + w) = v \cdot v + 2v \cdot w + w \cdot w = ||v||^2 + 2v \cdot w + ||w||^2.$$

Thus:

$$||v + w||^2 = ||v||^2 + ||w||^2 \iff ||v||^2 + 2v \cdot w + ||w||^2 = ||v||^2 + ||w||^2 \iff 2v \cdot w = 0.$$

Simplifying further:

$$2v \cdot w = 0 \iff v \cdot w = 0 \iff v \perp w.$$

Therefore, (II) $\iff$ (I).

We shall prove the equivalence (I) $\iff$ (III).

Starting:

$$\text{(III)} \iff ||v - w||^2 = ||v||^2 + ||w||^2.$$

Using the definition of the norm, we have:

$$||v - w||^2 = (v - w) \cdot (v - w) = v \cdot v - 2v \cdot w + w \cdot w = ||v||^2 - 2v \cdot w + ||w||^2.$$

Thus,

$$||v - w||^2 = ||v||^2 + ||w||^2 \iff ||v||^2 - 2v \cdot w + ||w||^2 = ||v||^2 + ||w||^2 \iff -2v \cdot w = 0.$$

Simplifying further:

$$-2v \cdot w = 0 \iff v \cdot w = 0 \iff v \perp w.$$

Therefore, (III) $\iff$ (I).

Since (I) $\iff$ (II) has already been proven, and we now have (I) $\iff$ (III), it follows that all the three statements are equivalent. □

Definition 2.4.8. A sequence $\{v_1, v_2, \ldots, v_s\}$ of vectors in the space $\mathbb{R}^n$ is called orthogonal if:

$$\forall i, j \in \{1, \ldots, s\},\ i \neq j \implies v_i \cdot v_j = 0.$$

Instead of saying "the sequence of vectors $\{v_1, v_2, \ldots, v_s\}$ is orthogonal", we could say "vectors $\{v_1, v_2, \ldots, v_s\}$ are pairwise orthogonal".

Theorem 2.4.9. *Let $w_1, w_2, \ldots, w_s \in \mathbb{R}^n$ be non-zero vectors. Assume that the sequence $\{w_1, w_2, \ldots, w_s\}$ is orthogonal. Then the vectors $w_1, w_2, \ldots, w_s$ are linearly independent.*

Proof. Let $\lambda_1, \ldots, \lambda_s \in \mathbb{R}^n$ be such that $\sum_{j=1}^{s} \lambda_j w_j = \overrightarrow{0}$

For any $i \in \{1, 2, \ldots, s\}$, since w_i is a non-zero vector, we have $||w_i|| \neq 0$. The sequence $\{w_1, w_2, \ldots, w_s\}$ is orthogonal, so for any $j \neq i$, we have:

$$w_i \cdot w_j = 0.$$

We now take the dot product of both sides of the equation $\sum_{j=1}^{s} \lambda_j w_j = \overrightarrow{0}$ with w_i:

$$0 = \left(\sum_{j=1}^{s} \lambda_j w_j \right) \cdot w_i = \sum_{j=1}^{s} \lambda_j (w_j \cdot w_i).$$

Since $w_j \cdot w_i = 0$ for $j \neq i$, only the term where $j = i$ remains:

$$0 = \lambda_i (w_i \cdot w_i) = \lambda_i |w_i|^2.$$

Since $w_i \neq \overrightarrow{0}$, we have $||w_i||^2 \neq 0$, so it follows that:

$$\lambda_i = 0.$$

Since this holds for any $i \in \{1, 2, \ldots, s\}$, we conclude that:

$$\lambda_1 = \lambda_2 = \cdots = \lambda_s = 0.$$

Thus, the vectors $w_1, w_2, \ldots, w_s$ are linearly independent, completing the proof.

□

Corollary

Let $w_1, \ldots, w_s \in \mathbb{R}^n$ be non-zero vectors. Then following statement is true:
The sequence of vectors $(w_1, \ldots, w_s)$ is orthogonal $\implies$ $s \leqslant n$.

2.5. Eigenvalues and eigenvectors

We will denote the set of all $m \times n$ matrices over the field $\mathbb{F}$ by $M_{m \times n}(\mathbb{F})$. We will write $M_n(\mathbb{F})$ instead of $M_{n \times n}(\mathbb{F})$. Characteristic polynomial of a given matrix, denoted as P_A ,is a basic algebraic concept. It is defined for a square matrix A by the following theorem.

Theorem 2.5.1. *Let $A \in M_n(\mathbb{C})$. Then $\det(A - xI_n) \in \mathbb{C}[x]$, where I_n is the unit matrix.*

Theorem 2.5.2. *Let $A \in M_n(\mathbb{C})$, then:*

(I) $deg(P_A) = n$,
(II) the leading coefficient of the polynomial P_A is equal to $(-1)^n$,
(III) the constant term of the polynomial P_A is equal to $\det(A)$,
(IV) the coefficient of x^{n-1} in the polynomial P_A is equal to $(-1)^{n-1} \operatorname{tr}(A)$.

Proof. We prove the properties of the characteristic polynomial described above:

(I) The determinant $\det(A - xI_n)$ is a polynomial in x. The highest degree term arises when all x's are selected from xI_n, giving x^n. Thus, $\deg(P_A) = n$.

(II) In fact, the leading term of $\det(A - xI_n)$ comes from $(-x)^n$, which contributes $(-1)^n$.

(III) Substituting $x = 0$ into $P_A(x) = \det(A - xI_n)$ gives $P_A(0) = \det(A)$.

(IV) The x^{n-1} term arises by selecting $n - 1$ diagonal elements from $-xI_n$ and 1 element from A, contributing $\sum_{i=1}^{n} (-1)^{n-1} a_{ii} = (-1)^{n-1} tr(A)$.

Thus, the properties are proven. □

Theorem 2.5.3. *If $A = [a_{ij}] \in M_n(\mathbb{C})$ is a triangular matrix, then $P_A(x) = \prod_{j=1}^{n} (a_{jj} - x)$.*

Proof. It is obvious, as the determinant of the triangular matrix is obtained by the multiplication of all diagonal elements of the matrix which in that particular example are received by subtracting variable x from all the diagonal elements of matrix A. □

Definition 2.5.4. Let $A = [a_{ij}] \in M_n(\mathbb{C})$. Eigenvalue of matrix A is defined as every complex root of characteristic polynomial of A. Equivalently, the number $\lambda \in \mathbb{C}$ is an eigenvalue of matrix A if, and only if
$det(A - \lambda I_n) = 0$.

Theorem 2.5.5. *Every matrix with complex elements has at least one eigenvalue*

Proof. The proof is a direct consequence of fundamental theorem of algebra which states: "[...] every non-constant single-variable polynomial with complex coefficients has at least one complex root." □

Having presented the key information on matrix eigenvalues, it is necessary to describe related topics that will enable the use of the eigenvalues themselves in an algorithmic application.

Definition 2.5.6. Let $A \in M_n(\mathbb{C})$. The spectrum of the matrix A is defined as the set of all its eigenvalues.

From now on, the spectrum of a matrix A will be denoted by $\sigma(A)$. Thus, $\sigma(A) = \{\lambda \in \mathbb{C} : \det(A - \lambda I_n) = 0\}$. On top of that it is clear that $\sigma(A) \neq \emptyset$.

Definition 2.5.7. Let $A \in M_n(\mathbb{C})$ and $\lambda \in \sigma(A)$. The algebraic multiplicity of the eigenvalue λ is defined as the multiplicity of λ as a root of the characteristic polynomial of the matrix A.

Theorem 2.5.8. *Let $A \in M_n(\mathbb{C})$. Then, $\sum_{\lambda \in \sigma(A)} m_A(\lambda) = n$ (in other words, the sum of the algebraic multiplicities of all eigenvalues of the matrix A is equal to the size of this matrix).*

Definition 2.5.9. Let $A \in M_n(\mathbb{C})$. An eigenvector of the matrix A is defined as any nonzero vector $v \in \mathbb{C}^n$ satisfying the condition $\exists \lambda \in \mathbb{C} : Av = \lambda v$.

Thus, one can say that an eigenvector of the matrix A is a nonzero vector in the space $\mathbb{C}^n$, which, when multiplied by the matrix, does not change its direction (although it may change its orientation or length).

Theorem 2.5.10. *Let $A \in M_n(\mathbb{C})$ and let v be an eigenvector of the matrix A. Then, there exists exactly one scalar $\lambda \in \mathbb{C}$ such that $Av = \lambda v$. Moreover, this scalar is an eigenvalue of the matrix A.*

Theorem 2.5.11. *Let $A \in M_n(\mathbb{C})$ and let $\lambda \in \sigma(A)$. Then, there exist infinitely many nonzero vectors $v \in \mathbb{C}^n$ satisfying the condition $Av = \lambda v$.*

As an immediate consequence of Theorems 2.0.30 and 2.0.31 we obtain the following characterization: if $A \in M_n(\mathbb{C})$ and $\lambda \in \mathbb{C}$, then λ is an eigenvalue of A if and only if where exists a nonzero vector $v \in \mathbb{C}^n$ satisfying $Av = \lambda v$.

Theorem 2.5.12. *All eigenvalues of any symmetric matrix with real entries are real numbers (i.e., the characteristic polynomial of such a matrix factors into linear terms over the field $\mathbb{R}$).*

Theorem 2.5.13. *Every symmetric matrix with real entries has an orthonormal eigenbasis (i.e., if the matrix is of size n, then it is possible to find n pairwise orthogonal eigenvectors of unit length).*

2.6. Graph Laplacians

The graph Laplacian is a crucial element of the whole process. This section uses [3] to introduce the Laplacian concept, for further details the paper of [2] is a main reference. It is required to obtain information about dataset from the graph and proceed further with the spectral reduction algorithms and clustering. We divide Laplacians into two main groups. The first one is considered as unnormalized whereas the second is normalized. On top of that it is important to mention, what the Laplacian matrix is. The characteristics of each group with examples are to be seen below.

In this subsection we always consider G to be an undirected, weighted graph characterized by a weight matrix $W = [w_{ij}]$, where $w_{ij} = w_{ji} \geqslant 0$.

Definition 2.6.1. The unnormalized graph Laplacian is defined as the matrix:

$$L = D - W,$$

where D is previously defined degree matrix and W is adjacency matrix of the graph. W can also be replaced with A(the adjacency matrix of a graph in case it is not weighted). The unnormalized graph Laplacian has the following properties:

1. For an arbitrary vector $v \in \mathbb{R}^n$ the following is true:

$$v^\intercal L v = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(v_i - v_j)^2$$

2. L is symmetric and positive semi-definite.

Proof. To prove property 1 we shall use the definition of elements of matrix W:

$$v^\intercal L v = v^\intercal D v - v^\intercal W v = \textstyle\sum_{i=1}^{n} d_i v_i^2 - \sum_{i,j=1}^{n} v_i v_j w_{ij} =$$
$$\frac{1}{2}\left(\textstyle\sum_{i=1}^{n} d_i v_i^2 - 2\sum_{i,j=1}^{n} v_i v_j w_{ij} + \sum_{j=1}^{n} d_j v_j^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(v_i - v_j)^2.$$

The symmetry of L is directly inherited from the symmetry of matrices D and W. Positive semi-definitness comes from property 1. □

It is important to note that the unnormalized graph Laplacian does not rely on the diagonal entries of the adjacency matrix W. Any adjacency matrix that matches W in all off-diagonal positions produces the same unnormalized graph Laplacian L. As a result, self-loops in the graph do not affect the corresponding graph Laplacian.

A complementary part of the Laplacian matrices section is an explanation of normalized graph's Laplacian. The definition we shall introduce is available in the literature [2].

Definition 2.6.2. The normalized graph Laplacian is defined as one of the the matrices:

$$L_{sym} = I - D^{-1/2} W D^{-1/2},$$

$$L_{rw} = I - D^{-1} W.$$

sym is an abbreviation of symmetrical and *rw* is the abbreviation of random walk.

Below, we summarize key properties of L_{sym} and L_{rw}. Main reference is [7].

1. For every $v \in \mathbb{R}^n$, the following holds:

$$v^\top L_{\text{sym}} v = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{v_i}{\sqrt{d_i}} - \frac{v_j}{\sqrt{d_j}} \right)^2.$$

2. λ is an eigenvalue of L_{rw} with eigenvector u if and only if 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 if and only if λ and u satisfy the generalized eigenvalue problem $Lu = \lambda Du$.
4. Both L_{sym} and L_{rw} are positive semi-definite, and their eigenvalues are real and non-negative.

2.7. Influence of the graph's Laplacian eigenvalues on the structure of the graph

This subsection will be mainly based on [1], [2]. All previously mentioned information have provided us with sufficient understanding of key elements of graph theory and linear algebra. It is now an appropriate moment to combine those two elements and learn how to read graph properties having calculated the eigenvalues of its Laplacian. By definition, L(G) is a positive semidefinite and therefore as stated in theorem 2.5.12. The eigenvalues are obtained in a form such that $0 \leqslant \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$ with possible repetitions with respect to their multiplicities. Moreover it also fulfils the theorem 2.5.13, and because of that it is always possible to obtain n pairwise orthogonal eigenvectors. As the multiplicities may occur the pairwise orthogonality can be achieved using Gram-Schmidt Process. It constructs an orthogonal basis from a set of linearly independent vectors.

Theorem 2.7.1. *Let $w^{(1)}, \ldots, w^{(s)} \in \mathbb{R}^n$ be linearly independent vectors. The recursive formula*

$$v^{(1)} = w^{(1)},$$

$$v^{(k)} = w^{(k)} - \sum_{i=1}^{k-1} \frac{w^{(k)} \circ v^{(i)}}{||v^{(i)}||^2} v^{(i)}, \quad \text{for } k \in \{2, \ldots, s\},$$

correctly defines an orthogonal sequence $v^{(1)}, \ldots, v^{(s)}$ of nonzero vectors in the space $\mathbb{R}^n$. Moreover, for each $k \in \{1, \ldots, s\}$, the following equality holds:

$$Span_{\mathbb{R}}(v^{(1)}, \ldots, v^{(k)}) = Span_{\mathbb{R}}(w^{(1)}, \ldots, w^{(k)}).$$

They crucial eigenvalue to start with is the second smallest eigenvalue λ_2, also known as algebraic connectivity. The smallest eigenvalue in our situation would be $\lambda = 0$, which is considered trivial. First, we establish results connecting λ_2 with the growth behaviour of the graph G. To continue it is needed to introduce freely chosen vertex $w \in V(G)$. The growth refers to the manner in which the cardinality of $B_k(w)$ denoted as b_k for $k = 0, 1, 2, \ldots$ increases as k becomes larger. A key finding demonstrates that the graph G exhibits exponential growth, specifically $b_k \geqslant \alpha^k$ where α represents a constant that is dependent on a lower bound derived from λ_2.

Lemma 2.7.2. $b_k - b_{k-1} > \frac{2\lambda_2}{n(\Delta+\lambda_2)}\Big[b_k(n-b_k) + b_{k-1}(n-b_{k-1})\Big]$,
where Δ denotes the maximal vertex degree of a graph.

Proof. As shown by Fiedler [1], the second smallest Laplacian eigenvalue λ_2 satisfies the following variational expression:

$$\lambda_2 = \min \frac{2n \sum_{u,v \in E}(x_u - x_v)^2}{\sum_{u \in V} \sum_{v \in V}(x_u - x_v)^2}, \tag{2.3}$$

where the minimum is taken over all non-constant vectors $\mathbf{x} = (x_v)_{v \in V} \in \mathbf{l}^2(V(G)) \approx \mathbb{R}^n$, where $n = |V(G)|$.

To leverage this formula, let us choose $\mathbf{x}$ as follows:

$$x_v = \begin{cases} 1, & v \in B_{k-1}, \\ 0, & v \in B_k \setminus B_{k-1}, \\ -1, & v \notin B_k. \end{cases}$$

Substituting this into the variational formula based on **(I)** gives the inequality:

$$n(e_{k-1} + e_k) \geqslant \lambda_2\Big[b_{k-1}n_k + (n - b_k)n_k + 4b_{k-1}(n - b_k)\Big],$$

where $n_k = b_k - b_{k-1}$. Since $e_{k-1} + e_j \leqslant \Delta n_k$, further simplifications yield:

$$\frac{\Delta}{\lambda_2} n n_k \geqslant 2b_k(n - b_k) + 2b_{k-1}(n - b_{k-1}) - n_k(n - n_k) > 2\big[b_k(n - b_k) + b_{k-1}(n - b_{k-1})\big] - n n_k$$

This inequality describes how b_k and b_{k-1} grow relative to the graph's size (n) and connectivity (λ_2) thus proving the lemma. □

By replacing b_k with a continuous function $y(k)$, where $k \in \mathbb{R}^+$, and b_{k-1} with $y(k - dk)$, we derive a differential inequality for y. This leads to the solution:

$$y(t) = \frac{n}{1 + (n-1)e^{-\beta t}}, \tag{2.4}$$

where $\beta = \frac{4\lambda_2}{\Delta + \lambda_2}$. This satisfies the boundary conditions:

$$y' = \frac{\beta}{n} y(n - y), \text{ where } y(0) = 1 \tag{2.5}$$

At integer points $k > 1$, this function $y(k)$ dominates the sequence $\{b_k\}$, establishing the desired relationship. Which demonstrates how the size of the ball B_k grows over the time (t approximating k). Initially, growth is rapid as vertices are added exponentially. Eventually growth slows due to saturation effects $y(t) \longrightarrow n$

Having provided an explanation to the relation of a graph structure it's lower bound and λ_2, it would be unjustifiable not to provide same for the upper bound, hence:

Lemma 2.7.3. *Let G be a connected graph and let y(t) be the recently proposed solution (2.5) of (2.6) then:*

$$k > 1 \wedge b_k \leqslant \tfrac{n}{2} \longrightarrow b_k > y(k)$$

Theorem 2.7.4. *The eigenvalue λ_2 imposes an upper bound on the diameter of G:*

$$diam(G) \leqslant \frac{\Delta + \lambda_2}{4\lambda_2} \ln(n - 1). \tag{2.6}$$

Proof. To prove the theorem, it suffices to demonstrate that for $k \geqslant \frac{1}{\beta} ln(n - 1.$, the quantity $b_k = b_k(w)$ satisfies $b_k > \frac{n}{2}$ for any choice of w. Suppose, for the sake of contradiction, that $b_k \leqslant \frac{n}{2}$. By Lemma (2.5), we know that

$$b_k > y(k) \geqslant y\frac{1}{\beta} \ln(n - 1) = \frac{n}{2}.$$

This leads to a contradiction, completing the proof. □

2.8. Algorithms overview

In this subsection we shall revise three most common algorithms used for spectral clustering. Two of them will be using graph Laplacian, normalized and unnormalized. The remaining one is the pure k-means algorithm which is capable of performing the clustering on the multidimensional data sets, thus it can be considered as spectral clustering algorithm, intuitively not the most optimal when faced with complex dataset. It is a great reference point for the more sophisticated approaches to compare them against it. On top of that we shall incorporate it into the final steps of more sophisticated algorithms to emphasize the difference obtained only from Laplacian related calculation, not the final cluster point assignment. The k-means algorithm is designed to partition a dataset $\{x_1, x_2, \ldots, x_n\}$ into k clusters $\{C_1, C_2, \ldots, C_k\}$ in a way that minimizes the sum of squared distances between data points and their assigned centroids. This process ensures that the clusters are compact and distinct from one another.

We suppose our dataset consists of n "points" $x_1, \ldots, x_n$, which can represent arbitrary objects. The pairwise similarities $s_{ij} = s(x_i, x_j)$ between these points are quantified using a similarity function that is symmetric and non-negative. The corresponding similarity matrix is denoted by $S = [s_{ij}]$. The goal of the k-means algorithm is to minimize the cost function, also known as inertia, which measures the total within-cluster variance. The cost function is defined as:

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - c_j\|^2. \tag{2.7}$$

By iteratively updating the clusters and centroids, the algorithm ensures that this cost function decreases monotonically. For all the experiments to maintain unity in the methodology for further comparison between the algorithms the similarity matrix shall be constructed always with the same method chosen in the next section, which can be considered as a step 0 for each of the further presented algorithms.

k-means algorithm

Output: Similarity matrix $S \in M_n(\mathbb{R})$, an integer k, which represents the number of clusters to be formed.

0. Randomly place k centroids in the data space. These centroids serve as the initial estimates for the centres of the clusters.
1. Assign each data point x_i to the cluster whose centroid is closest, based on the squared Euclidean distance. The assignment rule is:

$$C_j = \{x_i : \|x_i - c_j\|^2 \leqslant \|x_i - c_\ell\|^2, \ \forall\ \ell = 1, 2, \ldots, k\}.$$

2. Recalculate the centroid of each cluster as the mean of all data points assigned to that cluster:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i.$$

3. Repeat the assignment and update steps until the centroids stabilize, meaning they no longer change significantly between iterations.
4. The algorithm stops when the centroids converge, that is, when the updates result in negligible changes in their positions or when a predefined maximum number of iterations is reached.

Output: Clusters $C_1, \ldots, C_k$.

Another approach to clustering employs the unnormalized graph Laplacian, a widely used construct in spectral methods. This algorithm utilizes the fundamental properties of the graph Laplacian to segment data into clusters, offering a straightforward yet effective way to analyze the underlying structure of a similarity graph. While computationally less sophisticated than its normalized counterparts, the unnormalized approach remains a foundational technique, particularly in scenarios where simplicity and direct implementation are prioritized.

This method represents the most basic form of spectral clustering using the graph Laplacian. By constructing the Laplacian directly from the similarity matrix, the algorithm ensures a direct correspondence between the graph's structure and the resulting clusters. Its reliance on the eigen decomposition of the Laplacian underscores the importance of spectral properties in revealing inherent data groupings, providing a reliable baseline for comparisons or extensions involving more complex methods.

Unnormalized spectral clustering algorithm from [3]

Input: Similarity matrix $S \in M_n(\mathbb{R})$, an integer k, which represents the number of clusters to be formed.

1. Compute the unnormalized Laplacian L.
2. Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.
3. Form the matrix $U \in M_{n \times k}(\mathbb{R})$, where the columns are the eigenvectors $u_1, \ldots, u_k$.
4. For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the$i$-th row of U.
5. Cluster the points $y_1, \ldots, y_n$ into k clusters $C_1, \ldots, C_k$ using the k-means algorithm.

Output: Clusters $A_1, \ldots, A_k$, where $A_i = \{j \mid y_j \in C_i\}$.

Last algorithm computes the generalized eigenvectors of the Laplacian by using L_{rw}, hence can be called "normalized".

Normalized spectral clustering algorithm from [4]

Input: Similarity matrix $S \in M_n(\mathbb{R})$, an integer k, which represents the number of clusters to be formed.

1. Compute the unnormalized Laplacian L.
2. Solve the generalized eigenproblem $Lu = \lambda Du$ to obtain the first k generalized eigenvectors $u_1, \ldots, u_k$.
3. Form the matrix $U \in M_{n \times k}(\mathbb{R})$, where the columns are the eigenvectors $u_1, \ldots, u_k$.
4. For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ represent the i-th row of U.
5. Cluster the points$y_1 \ldots, y_n$ into k clusters $C_1, \ldots, C_k$ using the k-means algorithm.

Output: Clusters $A_1, \ldots, A_k$, where $A_i = \{j \mid y_j \in C_i\}$.

The two algorithms discussed earlier are quite similar, with the primary difference being the type of graph Laplacian used in each. In both cases, the key idea is to transform the original data points x_i into a new representation $y_i \in \mathbb{R}^k$. This transformation leverages the properties of the graph Laplacians, making it effective. Specifically, the standard k-means clustering algorithm can easily identify the clusters in this transformed space.

3. Methodology

This chapter provides a comprehensive explanation and introduction to the program written in *Python programming language, version: 3.11.8*. The program presents an in-depth analysis of the spectral clustering algorithms described in the previous section, demonstrating their theoretical foundations and practical applications through carefully designed experiments. The implementation is structured across two dedicated Jupyter notebooks, each serving a distinct purpose in the research. The first notebook, titled "Research", offers a step-by-step walkthrough of the spectral clustering algorithm, examining its behavior and validating the theoretical concepts previously discussed. It provides a detailed analysis of the mathematical foundations behind spectral clustering, including similarity matrix construction, graph representation, eigenvalue decomposition, and clustering using k-means on transformed data. Additionally, it features practical implementations in Python, allowing for verification of theoretical insights through numerical experiments. These experiments utilize the widely recognized IRIS dataset, a well-known benchmark dataset in machine learning, which facilitates the demonstration of key clustering properties and algorithmic behaviour. The notebook also includes various visualization techniques to illustrate how spectral clustering segments data points based on their intrinsic structure. The second notebook titled "Comparison" is entirely dedicated to evaluating the usability and effectiveness of spectral clustering in practical scenarios. In this notebook, the clustering algorithms are applied not only to the IRIS dataset but also to a more complex real-world dataset: the "ECG of Cardiac Ailments Dataset". This dataset contains real patient electrocardiogram (ECG) recordings, providing a more challenging and diverse dataset for clustering analysis. Since both datasets contain ground truth labels, the quality of clustering is assessed by comparing the generated clusters to the original class labels. This evaluation is conducted using standard clustering quality metrics, such as Adjusted Rand Index (ARI), to quantify the accuracy of the clustering results. The notebook also discusses computational performance, considering runtime and memory usage, to analyse the scalability of spectral clustering when applied to a larger dataset with higher dimensionality. Both Jupyter notebooks, along with their corresponding code implementations and additional resources, are publicly available in a GitHub repository. The repository can be accessed at the following link: *https://github.com/KubaPtak/SpectralClustering*

3.1. Resources

The research conducted in this study is based on two publicly available datasets, sourced from the Kaggle platform and directly from UCI Machine Learning Repository. These datasets serve as the foundation for analyzing and evaluating the performance of spectral clustering algorithms under different conditions. The first dataset, IRIS, is a widely recognized benchmark dataset frequently used in machine learning and statistical classification tasks. It consists of 150 samples from three different species of the Iris flower—Setosa, Versicolor, and Virginica—each described by four numerical features: sepal length, sepal width, petal length, and petal width. Due to its well-structured nature and relatively small size, the dataset is particularly useful for demonstrating the fundamental principles of clustering algorithms. The dataset can be accessed at the following link: *https://archive.ics.uci.edu/dataset/53/iris*. The second dataset, titled "ECG of Cardiac Ailments Dataset", presents a more complex real-world dataset containing electrocardiogram (ECG) readings of patients with various heart conditions. Unlike the IRIS dataset, this dataset introduces additional challenges due to its higher dimensionality and inherent noise.

By incorporating this dataset into the study, the goal is to assess how spectral clustering techniques perform in more practical, dimensionally extensive conditions. The dataset is available at: *https://www.kaggle.com/datasets/akki2703/ecg-of-cardiac-ailments-dataset*. Both datasets were downloaded and archived in the project's GitHub repository to ensure reproducibility, with the download date recorded as 10th January 2025. To process and analyse the datasets, a variety of scientific computing libraries have been used. Core mathematical operations, including matrix computations and similarity graph constructions, were performed using *NumPy 2.2.2*, while data extraction, preprocessing, and transformation tasks were handled with *Pandas 2.2.3*. Additionally, mathematical functions were implemented using *SciPy 1.15.1*. For clustering model development and evaluation, the study employs *scikit-learn 1.6.1*, which provides a robust implementation of the spectral clustering algorithm, along with essential tools for performance assessment. The visualization of clustering results, including data distributions and cluster separations, was carried out using *Seaborn 0.12.2*, ensuring clear and interpretable graphical representations.

3.2. IRIS Dataset

The IRIS dataset consists of 150 observations with five attributes: `sepal_length`, `sepal_width`, `petal_length`, `petal_width`, and `species`. The first four attributes are continuous numerical features, while the `species` column is categorical. Below is the dataset summary with the result of the output of the code visible on Figure 3.1:

```python
# Display dataset information
print("\nDataset Information:")
print(df.info())

# Display first few rows
print("\nFirst Five Rows of the Dataset:")
print(df.head())

# Display summary statistics
print("\nSummary Statistics:")
print(df.describe())

# Species distribution
print("\nSpecies Count:")
print(df.iloc[:, -1].value_counts())

# Set style for seaborn plots
sns.set(style="whitegrid")

# Plot histograms for all numerical features
plt.figure(figsize=(12, 8))
df.iloc[:, :-1].hist(bins=20, figsize=(12, 8), layout=(2, 2), color='skyblue', edgecolor='black')

# Show all plots in one window
plt.show()
```

Figure 3.1: Research file code used to obtain information about IRIS dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
```

```
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

This confirms that there are no missing values in the dataset, and the features are stored as floating-point numbers.

The first five entries of the dataset are displayed in Table 3.1.

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa

Table 3.1: First five rows of the IRIS dataset.

Table 3.2 provides descriptive statistics for the numerical features. The table highlights that

Statistic	sepal_length	sepal_width	petal_length	petal_width
Count	150.0	150.0	150.0	150.0
Mean	5.843	3.054	3.759	1.199
Std Dev	0.828	0.434	1.764	0.763
Min	4.3	2.0	1.0	0.1
25%	5.1	2.8	1.6	0.3
50% (Median)	5.80	3.00	4.35	1.30
75%	6.4	3.3	5.1	1.8
Max	7.9	4.4	6.9	2.5

Table 3.2: Summary statistics of IRIS dataset.

sepal length varies from 4.3 cm to 7.9 cm, while petal width has a maximum of 2.5 cm and is relatively less variable compared to sepal length.

3.3. Species Distribution

The dataset contains three species, each with 50 samples:

— **Iris-setosa**: 50 samples
— **Iris-versicolor**: 50 samples
— **Iris-virginica**: 50 samples

This balanced distribution ensures an even representation of all species in the dataset.

The histograms for each numerical feature are shown in Figure 3.2:

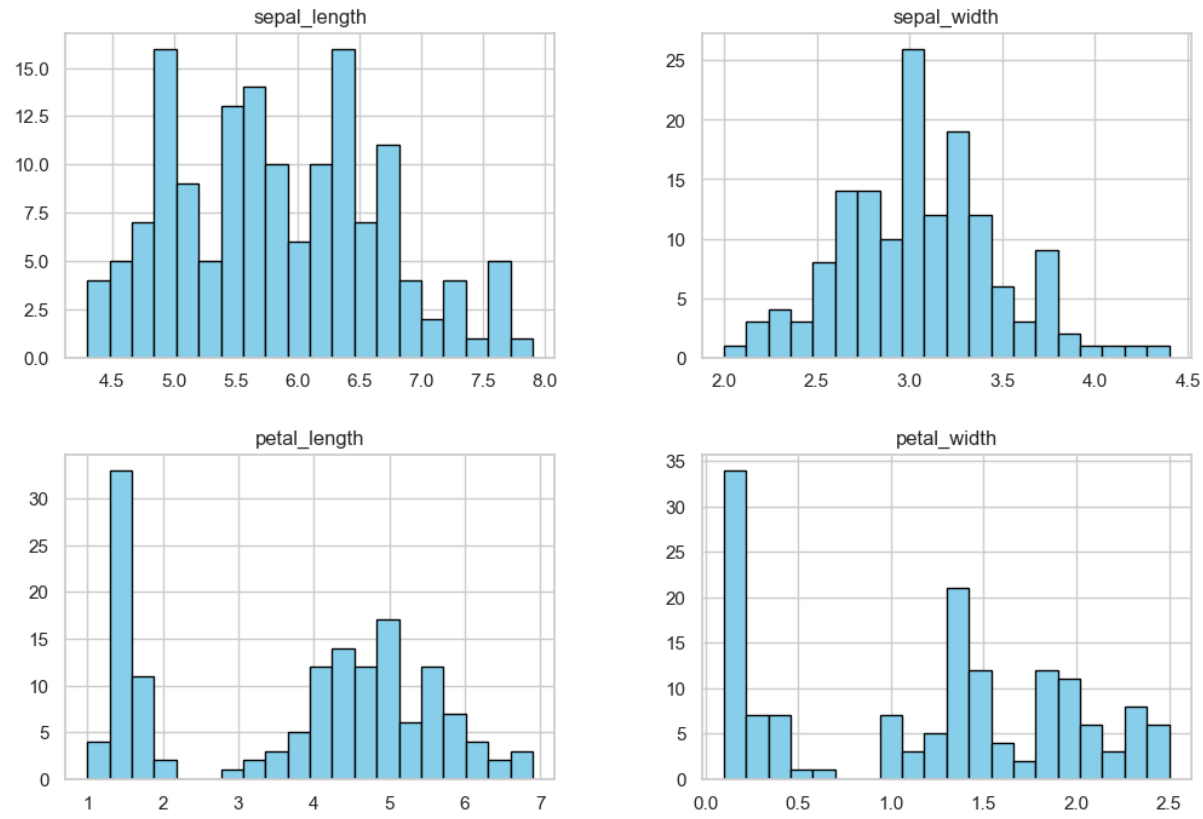


Figure 3.2: Distribution of numerical features in the IRIS dataset

These histograms illustrate the spread of sepal and petal measurements, providing insights into feature separability.

3.4. ECG Dataset

The dataset consists of 1200 entries, with 56 columns, containg both numerical and categorical data. The dataset's summary statistics are provided below for the first 7 variables:

Variable	Mean	Standard Deviation	Min	Max
hbpermin	81.89	19.32	12.86	160.50
Pseg	0.0609	0.00945	0.0216	0.0953
PQseg	0.0781	0.0199	0.0445	0.1456
QRSseg	0.0482	0.0346	0.0000	0.1202
QRseg	0.0244	0.0175	0.0000	0.0653
QTseg	0.1415	0.0292	0.0988	0.2111
RSseg	0.0238	0.0172	0.0000	0.0583

The dataset contains ECG signals labeled by categories. The class distribution is as follows:

ECG Signal Class	Count
NSR	300
ARR	300
AFF	300
CHF	300

The dataset consists of 1200 ECG records with 56 features, including time-domain, frequency-domain, and morphological features of the ECG signals. It is well-balanced across four classes: NSR, ARR,

AFF, and CHF. Some features contain NaaN values. They are replaced with the average result for a given signal class.

```
columns_to_fill = ['QRtoQSdur', 'RStoQSdur', 'PonPQang', 'PQRang', 'QRSang', 'RSTang', 'STToffang', 'QRslope', 'RSslope']

# Fill NaN values with the mean of the corresponding ECG_signal group
df[columns_to_fill] = df.groupby('ECG_signal')[columns_to_fill].transform(lambda x: x.fillna(x.mean()))
```

Figure 3.3: Comparison file code used to fill NaaN values with numerical values.

3.5. "Research" Notebook

This subsection contains crucial information verified in the "Research" notebook. It mainly focuses on the IRIS dataset as any further operations performed on ECG dataset are based on the same logic. Having discussed the specification of the datasets, we shall take into consideration the information obtained from them. First thing to consider is the inertia for k-means algorithm used on IRIS dataset.

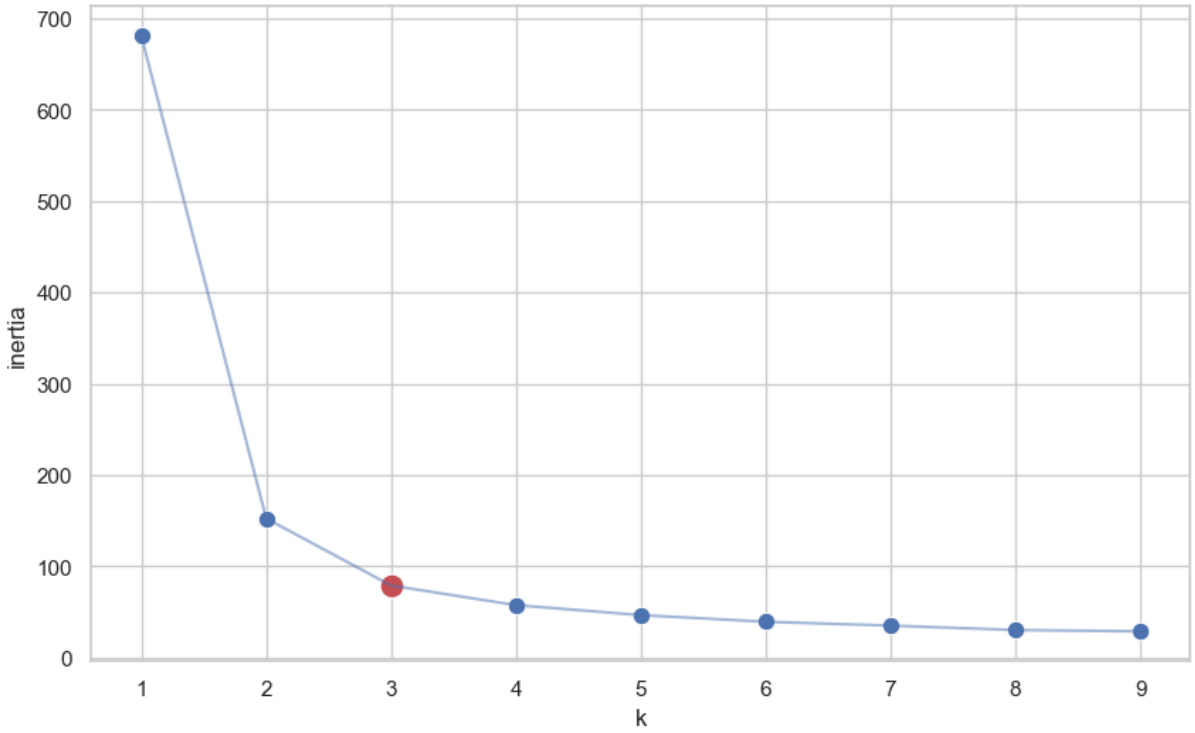


Figure 3.4: Inertia chart from "Research" notebook calculated for k-means algorithm on IRIS dataset.

The inertia has been checked to see if it is similar to the native number of classes of the dataset. It is clearly visible from the 2.7 that it indicates the most optimal number of clusters as 3, which is exactly the number of species in the dataset. To continue further it is required to introduce method for creating similarity matrix out of the dataset. For that we would use k-nearest neighbours algorithm. The graph Laplacians, both standard and normalized, look as follows:

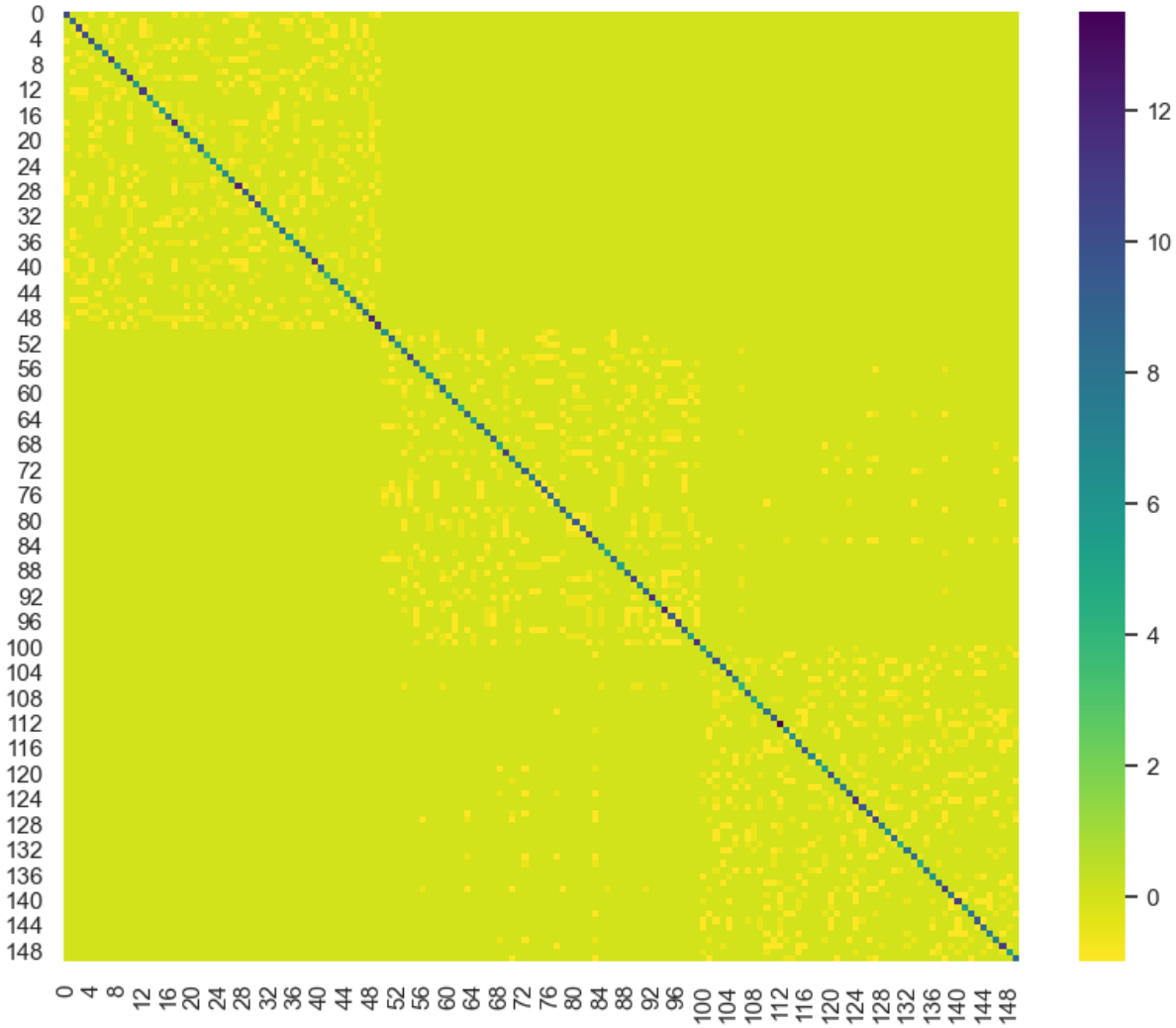


Figure 3.5: Unnormalized Laplacian for IRIS dataset, calculated in "Research" notebook.

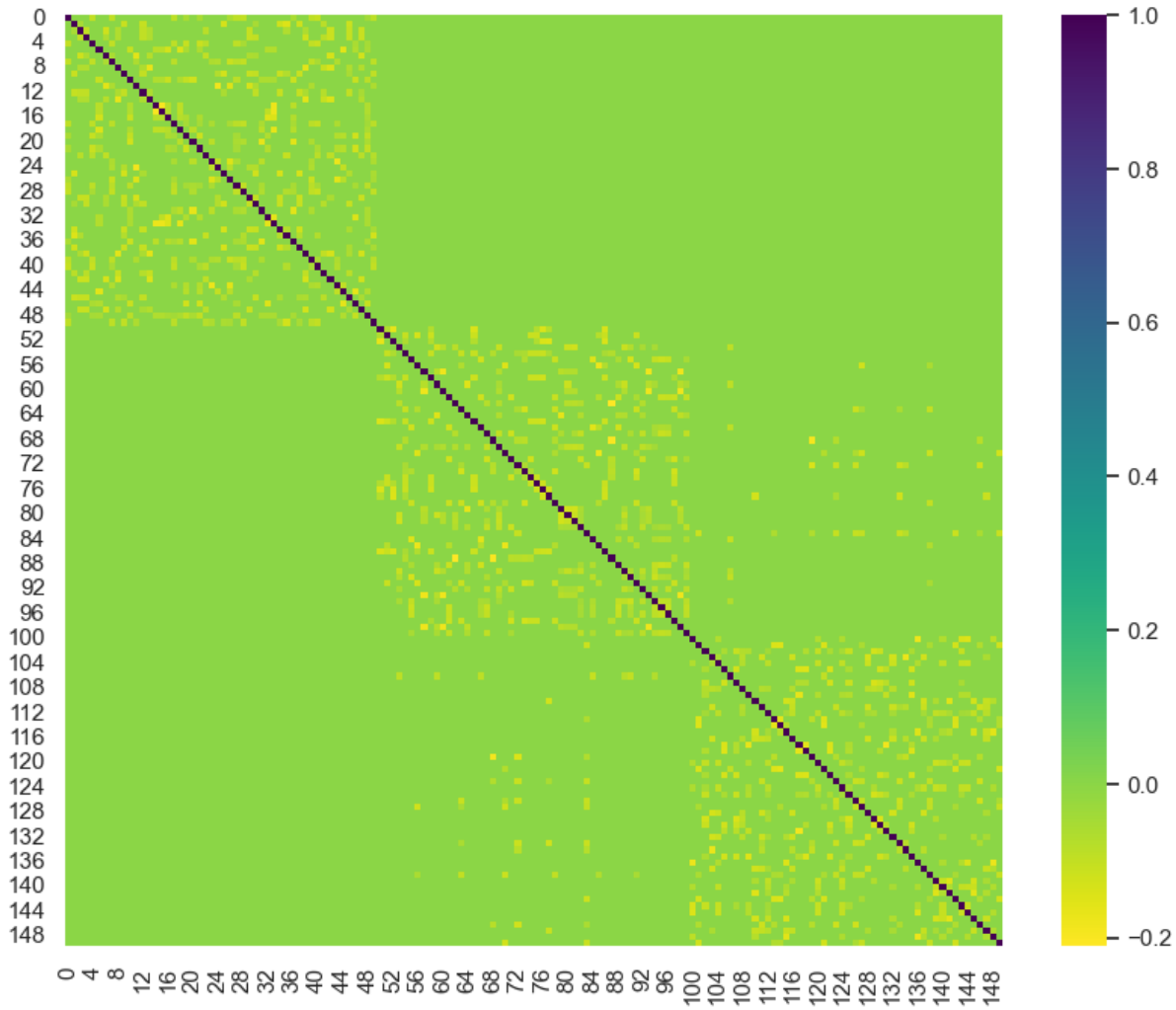


Figure 3.6: Normalized Laplacian for IRIS dataset, calculated in "Research" notebook.

Next step would be a cross-check for the reality of the normalized and unnormalized Laplacians' eigenvalues as per theorem 2.5.12.

```
from scipy import linalg

eigenvals, eigenvcts = linalg.eig(graph_laplacian)
eigenvals_norm, eigenvcts_norm = linalg.eig(graph_laplacian_norm)
np.unique(np.imag(eigenvals))
```

Figure 3.7: Lines of code responsible for obtaining eigenvalues and eigenvectors of normalized and un-normalized Laplacians of IRIS dataset and then verifying eigenvalues' reality, calculated in "Research" notebook.

The output of the 3.7 is an empty array thus the calculations are aligned with theorem 2.5.12. Next point would be having a perspective on sorted eigenvalues of both Laplacian matrices.

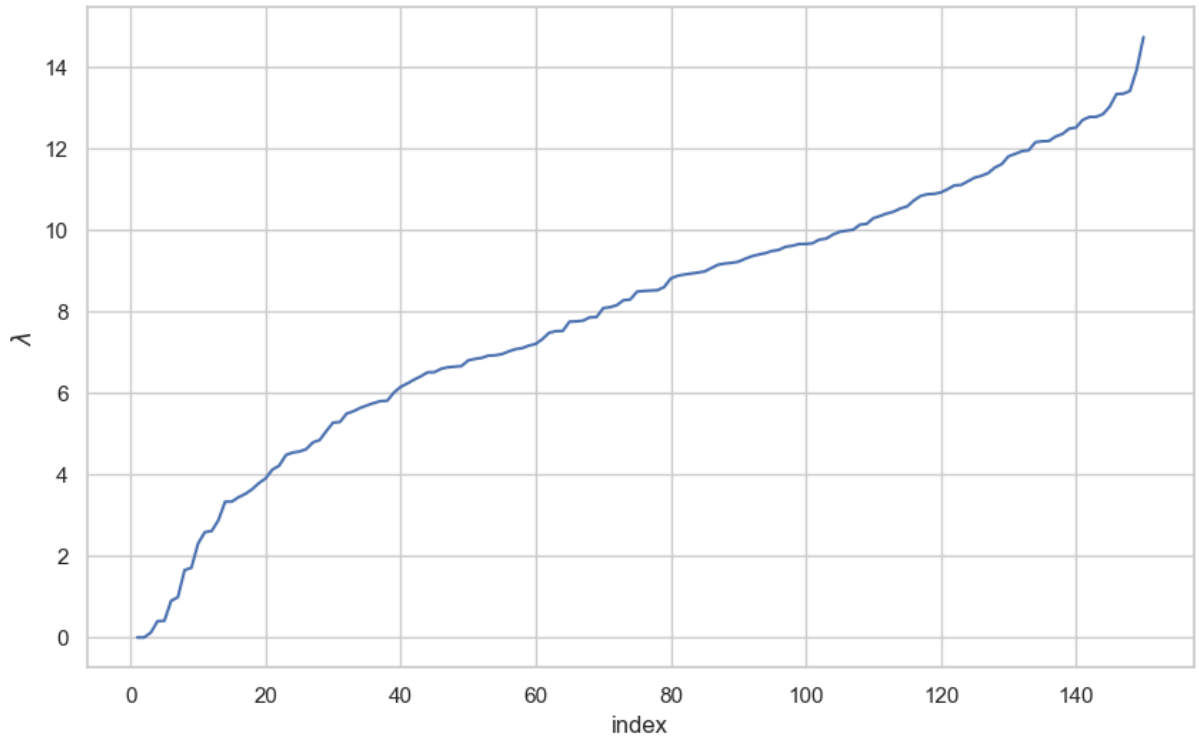


Figure 3.8: Sorted eigenvalues of un-normalized Laplacian, calculated in "Research" notebook.

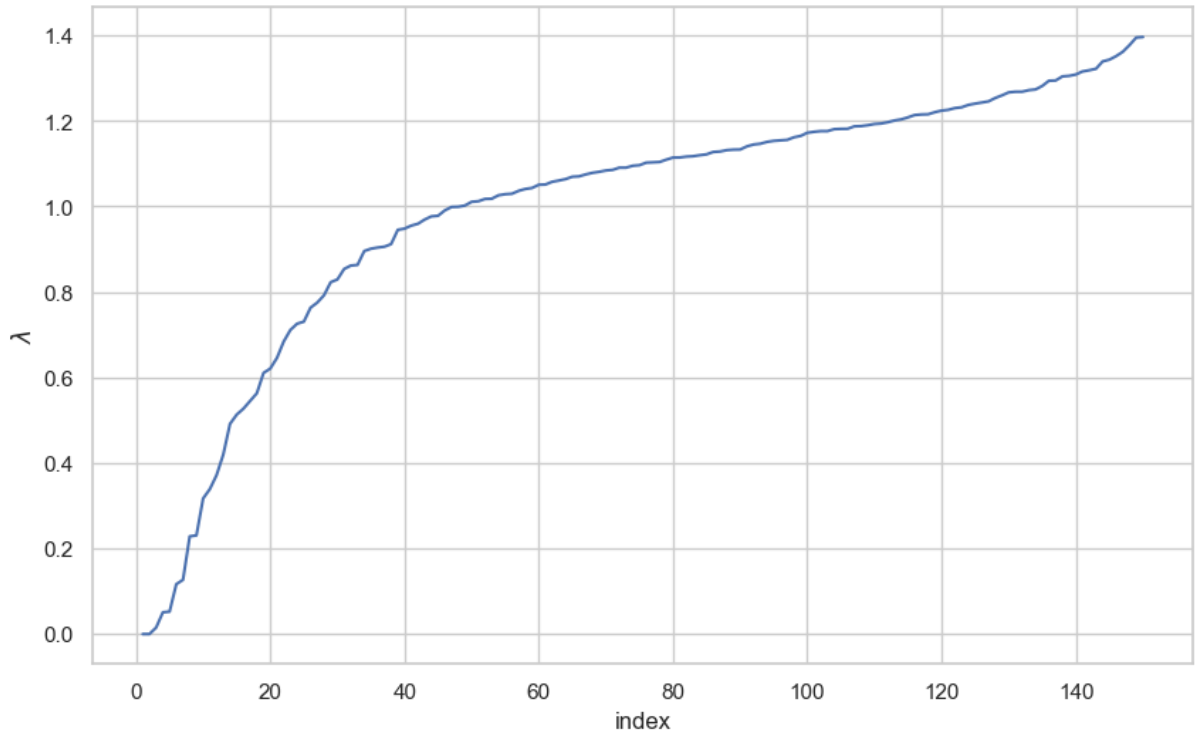


Figure 3.9: Sorted eigenvalues of normalized Laplacian, calculated in Research notebook.

Choosing first three smallest eigenvalues to further calculate eigenvectors with, we are coming to the next step itself. Here is how it looks like for standard graph Laplacian.

```
array([[-6.66133815e-16],
       [ 1.47629805e-15],
       [ 1.20652016e-01]])
```

Figure 3.10: Three smallest non-zero eigenvalues for unnormalized graph Laplacian for IRIS dataset, calculated in Research notebook.

	v_0	v_1	v_2
0	-0.141421	0.0	0.000000
1	-0.141421	0.0	0.000000
2	-0.141421	0.0	0.000000
3	-0.141421	0.0	0.000000
4	-0.141421	0.0	0.000000
...	...	...	...
145	0.000000	-0.1	-0.109345
146	0.000000	-0.1	0.005521
147	0.000000	-0.1	-0.092532
148	0.000000	-0.1	-0.114791
149	0.000000	-0.1	0.024145

Figure 3.11: Matrix obtained by selecting eigenvectors obtained from eigenvalues from 3.10, calculated in "Research" notebook.

Figure 3.12 shows the milestone in every spectral clustering algorithm. The dimensionality of the data has been reduced to just three columns. Now the standard k-means cluster assignment algorithm can be performed in the given, dimensionally reduced dataframe.

3.6. "Comparison" Notebook

The purpose of this notebook is to collect all the previous steps from "Research" notebook and create tested spectral clustering for IRIS and ECG datasets and compare the results afterwards. The results of the clustering experiments on the IRIS and ECG datasets are always performed assuming the number of calculated clusters is always equal to the number of classes in the source dataset, which is three for IRIS dataset and four for ECG data source. They demonstrate key differences in the performance of k-means, standard spectral clustering, and normalized spectral clustering. For measuring efficiency and quality we need to introduce The Adjusted Rand Index (ARI) which measures clustering similarity with a ground truth partition, adjusted for chance. Given n samples, let a be the number of element pairs in the same cluster in both true and predicted partitions, and b be the number of pairs in different clusters in both. The Rand Index (RI) is:

$$RI = \frac{a+b}{\binom{n}{2}}. \tag{3.1}$$

ARI adjusts RI to correct for randomness:

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]}. \tag{3.2}$$

It ranges from -1 (worse than random) to 1 (perfect match). The Silhouette Score evaluates cluster compactness and separation. For a point i, define:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}, \tag{3.3}$$

where $a(i)$ is the average intra-cluster distance, and $b(i)$ is the nearest-cluster distance. The overall score is the mean of all $s(i)$, ranging from -1 (poor clustering) to 1 (well-separated clusters). Based on the results of the experiment presented in Table 3.3, for the IRIS dataset, which is relatively small and well-structured, all three algorithms performed comparably in terms of clustering quality. The Adjusted Rand Index (ARI) for k-means was 0.73, while both spectral clustering methods achieved a slightly higher ARI of 0.76. The average Silhouette Score remained consistent across all methods at 0.55. However, in terms of computational efficiency, k-means was the fastest, completing in 0.2142 seconds with minimal memory usage (0.25 MB). Standard spectral clustering required 0.2596 seconds and 0.36 MB of memory, while the normalized spectral clustering method had a similar runtime (0.2594 seconds) but significantly higher memory consumption (1.79 MB). This suggests that for small, well-structured datasets, k-means provides an efficient and effective clustering solution. For the ECG dataset, which is more complex, the differences between the algorithms became more pronounced. The k-means algorithm exhibited a much lower ARI of 0.37, indicating poor cluster separation, though it remained the fastest method with a runtime of 0.2141 seconds and a memory usage of 0.67 MB. Standard spectral clustering improved clustering performance significantly, achieving an ARI of 0.59, though at the cost of increased computational time (1.4772 seconds) and much higher memory usage (23.27 MB). The normalized spectral clustering algorithm further improved clustering performance, reaching the highest ARI of 0.68. However, it also required substantial computational resources, with a runtime of 1.4159 seconds and memory usage of 22.21 MB. These results highlight the advantage of normalized spectral clustering for complex datasets. While k-means is an efficient option for simpler datasets that can be processed on local machines with minimal computational resources, it struggles with more complex data structures. Standard spectral clustering provides an improvement, but normalized spectral clustering demonstrates its superiority in handling intricate datasets by yielding the highest clustering quality, as indicated by the ARI scores. However, this comes at the cost of significantly increased memory consumption, which may limit its applicability to large-scale problems without access to high-performance computing resources.

Overall, the choice of clustering algorithm depends on the complexity of the dataset and the available computational resources. For small, well-separated datasets, k-means remains a viable option due to its simplicity and speed. However, for datasets with more complex structures, normalized spectral clustering proves to be the most effective approach despite its higher computational cost.

Algorithm	ARI	Silhouette Score	Clustering Time (sec)	Memory Used (MB)
k-means (Iris)	0.73	0.55	0.2142	0.25
Standard Laplacian Spectral Clustering (Iris)	0.76	0.55	0.2596	0.36
Normalized Laplacian Spectral Clustering (Iris)	0.76	0.55	0.2594	1.79
k-means (ECG)	0.37	0.37	0.2141	0.67
Standard Laplacian Spectral Clustering (ECG)	0.59	0.27	1.4772	23.27
Normalized Laplacian Spectral Clustering (ECG)	0.68	0.31	1.4159	22.21

Table 3.3: Clustering Results for IRIS and ECG Datasets

```python
from sklearn.neighbors import kneighbors_graph
from scipy import sparse
from scipy import linalg

def spectral_clustering_analysis(df, Y, nn=8, n_clusters=3, random_state=25):
    start_time = time.time()
    mem_before = memory_usage()[0]
    # Generate graph Laplacian
    connectivity = kneighbors_graph(X=df, n_neighbors=nn, mode='connectivity')
    adjacency_matrix_s = (1/2)*(connectivity + connectivity.T)
    graph_laplacian_s = sparse.csgraph.laplacian(csgraph=adjacency_matrix_s, normed=True)
    graph_laplacian = graph_laplacian_s.toarray()
    # Compute eigenvalues and eigenvectors
    eigenvals, eigenvcts = linalg.eig(graph_laplacian)
    eigenvals = np.real(eigenvals)
    eigenvcts = np.real(eigenvcts)
    # Project onto first few eigenvectors
    eigenvals_sorted_indices = np.argsort(eigenvals)
    indices = eigenvals_sorted_indices[:3]
    proj_df = pd.DataFrame(eigenvcts[:, indices.squeeze()])
    proj_df.columns = ['v_' + str(c) for c in proj_df.columns]
    # Perform K-Means clustering
    k_means = KMeans(random_state=random_state, n_clusters=n_clusters)
    k_means.fit(proj_df)
    cluster = k_means.predict(proj_df)
    mem_after = memory_usage()[0]  # Memory after computations
    elapsed_time = time.time() - start_time
    return cluster, elapsed_time, mem_after - mem_before


cluster, elapsed_time, memory_diff = spectral_clustering_analysis(X, Y, nn=8, n_clusters=3, random_state=25)
```

Figure 3.12: A function used to perform normalized spectral clustering and measure its efficiency, calculated in "Comparison" notebook.

4. Conclusion

In this study, we analyzed the performance of spectral clustering algorithms—both standard and normalized Laplacian-based—alongside the k-means algorithm as a baseline. Our results highlight the advantages and limitations of these approaches, particularly in the context of increasing data complexity. For simpler datasets, such as the IRIS dataset, k-means proves to be the most efficient choice, offering the fastest computation time with a relatively high clustering quality. However, as dataset complexity increases, as seen with the ECG dataset, the limitations of k-means become evident, particularly in terms of clustering accuracy. The spectral clustering approaches demonstrate superior performance in these cases, with the normalized Laplacian-based method achieving the highest Adjusted Rand Index (ARI). This result aligns with theoretical expectations, as the normalized Laplacian provides better stability and robustness in cases of imbalanced or high-dimensional data. Despite their advantages in clustering quality, spectral clustering methods exhibit significantly higher computational costs, particularly in terms of memory usage. This trade-off suggests that while k-means remains a viable option for lower-dimensional problems that can be efficiently computed on local machines, spectral clustering—especially the normalized variant—is more suitable for complex, high-dimensional datasets where accuracy is the primary concern.

Bibliography

[1] M. Fiedler, *Eigenvalues and graph theory*, Czechoslovak Mathematical Journal, 1965.

[2] B. Mohar, *Properties of Laplacian eigenvalues in graph theory*, Discrete Mathematics, 1991.

[3] U. von Luxburg, *A tutorial on spectral clustering*, arXiv:0711.0189v1 [cs.DS], 2007.

[4] Jianbo Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

[5] A. Y. Ng, M. I. Jordan, Y. Weiss, *On spectral clustering: analysis and an algorithm*, NIPS'01: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, 2001, pages: 849-856.

[6] Y. Wang, *Improving spectral clustering using spectrum-preserving node aggregation*, arXiv:2110.12328v6 [cs.LG], 2023.

[7] F. Chung, *Spectral graph theory*, Vol. 92 of the CBMS Regional Conference Series in Mathematics, Conference Board of the Mathematical Sciences, 1997.

[8] L. Alekhya and P. Rajesh Kumar, *A new approach to detect cardiovascular diseases using ECG scalograms and ML-based CNN algorithm*, International Journal of Computational Vision and Robotics, Mar 20, 2023, DOI: 10.1504/IJCVR.2022.10051429, https://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=IJCVR.

[9] L. Alekhya and P. Rajesh Kumar, *A Novel Application for Autonomous Detection of Cardiac Ailments using ECG Scalograms with Alex Net Convolution Neural Network*, Design Engineering, 2021, pp. 13176-13189, http://www.thedesignengineering.com/index.php/DE/article/view/6434.

[10] L. Alekhya, P. Rajesh Kumar, and A. Venkata Sriram, *Autonomous Detection of Cardiac Ailments using Long-short term Memory Model based on Electrocardiogram signals*, NeuroQuantology, 2022, DOI: 10.14704/nq.2022.20.7.NQ33431, pp. 3509-3518, https://www.neuroquantology.com/open-access/Autonomous+Detection+of+Cardiac+Ailments+using+Longshort+term+Memory+Model+based+on+Electrocardiogram+signals_5781/.

[11] L. Alekhya and P. Rajesh Kumar, *Autonomous Detection of Cardiac Ailments diagnosed by Electrocardiogram using various Supervised Machine Learning Algorithms*, AMA, Agricultural Mechanization in Asia, Africa and Latin America, Sep 18, 2021, https://www.shin-norinco.com/article/autonomous-detection-of-cardia-ailments-diagnosed-by-electrocardiogram-using-various-supervised-mac

[12] L. Alekhya and P. Rajesh Kumar, *Maximal Overlap Discrete Wavelet Packet Transform Based Characteristic waves detection in Electrocardiogram of Cardiovascular Diseases*, INTERNATIONAL JOURNAL OF SPECIAL EDUCATION, vol 36 (1), pp. 51-61, 2021.