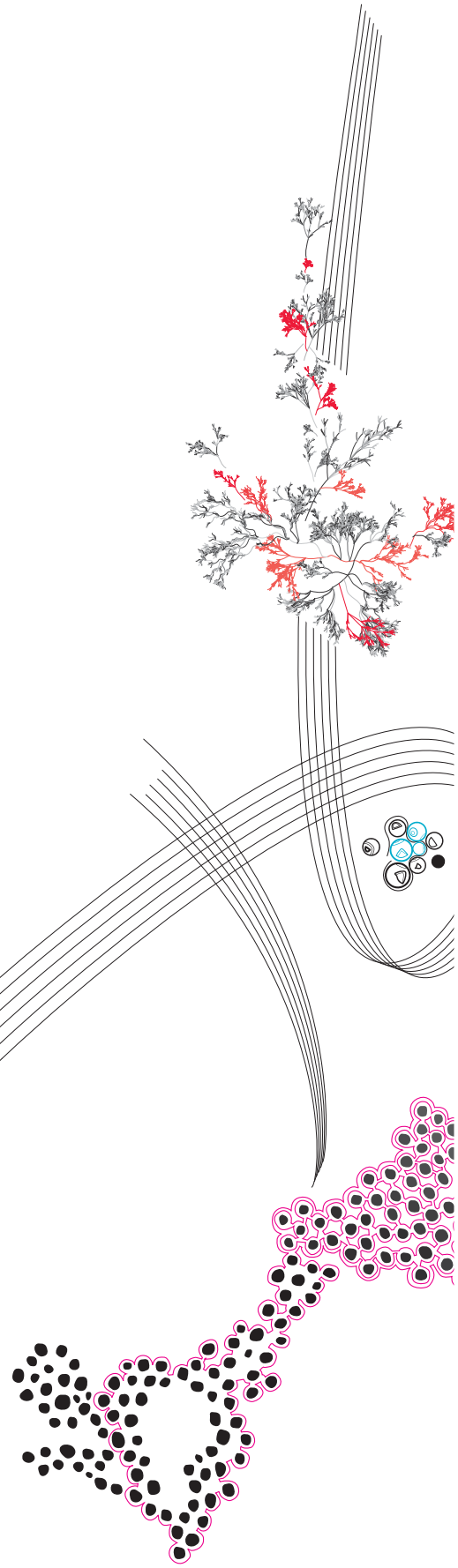

BSc Thesis Applied Mathematics

Accelerated Spectral Clustering using Random Forest and the Locally Linear Landmark approach

Zoë van Herreveld

Supervisors: Annika Betken and Hongwei Wen

August, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

This report is part of my Bachelor's assignment, which marks the completion of my Bachelor's degree in Applied Mathematics at the University of Twente. I would like to express my gratitude to Annika Betken and Hongwei Wen for their supervision and invaluable input throughout my research. I am very grateful that you took the time to join my research when unforeseen circumstances left me without supervisors, ensuring I could complete my Bachelor's assignment without further obstacles. Additionally, I would like to thank my family, friends, and housemates for their support not only during my research but throughout the entirety of my Bachelor's degree.

Accelerated Spectral Clustering using Random Forest and the Locally Linear Landmarks approach

Zoë van Herreveld*

August, 2024

Abstract

In the era of big data, where data streams are constantly expanding in all aspects of our life, advanced clustering techniques are in big demand. One popular clustering technique in the field of unsupervised learning is spectral clustering. This is a graph-based clustering method that, as the name already implies, is able to capture complex data structures by leveraging the spectral properties of the data. This involves analyzing the eigenvalues and eigenvectors of certain matrices. However, particularly due to this eigendecomposition, the computational cost of spectral clustering is very high, especially when dealing with large datasets. This research explores the potential of enhancing the complexity of spectral clustering by using extremely randomized trees to construct the similarity matrix and by implementing the Locally Linear Landmarks (LLL) technique. This is a dimensionality reduction technique, first introduced by M. Vladymyrov and M. À . Carreira-Perpiñàn in [19] for manifold learning. By replacing binary trees with extremely randomized trees, significant reductions in runtime are achieved without losing performance. This is demonstrated with various datasets. Additionally, our implementation of the LLL method yielded notable improvements in runtime, however with a notable trade-off in terms of performance in some cases. Lastly, this research contributes a successful Python implementation of the general spectral clustering algorithm. This implementation incorporates a random forest based similarity measure *RatioRF* [2] for calculating the similarity matrix and offers the option to use the Linear Locally Landmarks approach, paving the way for further research and advancements in the field of spectral clustering.

Keywords: Spectral Clustering, Extremely Randomized Trees, Random Forest, Locally Linear Landmarks

*Email: z.vanherreveld@student.utwente.nl

Contents

1 Introduction

In the era of big data, where data streams are constantly expanding in all aspects of our life, advanced data analysis techniques are in big demand. One such techniques that has been evolving over the past decade is clustering.

Clustering algorithms aim to group data points together that are more similar based on their characteristics or features. They usually achieve this by focusing on partitioning the data using geometric measures, like distance or density. However, recently there has been introduced a new similarity measure, called *RatioRF* [2], which assesses the similarity between data points based on their behavior within trained decision trees. Their use of binary decision trees to generate the *RatioRF* measure has already yielded promising results. Binary trees, however, can be computationally expensive due to their search for the optimal split at every node. A good alternative could be the use of extremely randomized trees. This type of decision trees selects features and splitting thresholds randomly, thereby avoiding the need for an exhaustive search to find the optimal split. This additional randomness leads to faster training times, while also reducing overfitting and increasing robustness to noisy data.

One popular clustering technique in the field of unsupervised learning is spectral clustering. While most clustering techniques focus on partitioning the data in the feature space based directly on the geometric arrangement of the data points, spectral clustering approaches it as a graph partitioning problem. Data points are represented as nodes in a graph, and their pairwise similarities are represented as edges with certain weights. This allows for capturing complex structures and patterns in the data that may not be visible in the feature space. As the name implies, spectral clustering achieves this by exploiting the spectral properties of the data. This involves analyzing the eigenvalues and eigenvectors of matrices derived from the data. One of the biggest bottlenecks of spectral clustering is its computational cost, particularly due to this eigendecomposition. One way to reduce this complexity is to perform a dimensionality reduction technique, such as the Locally Linear Landmarks (LLL) technique. This was first introduced by M. Vladymyrov and M. À . Carreira-Perpiñàn in [19] for manifold learning. The LLL method reformulates the spectral clustering problem to a reduced problem, which makes the eigendecomposition in the process more efficient, since it will be done on smaller matrices. It achieves this by introducing a number of landmarks, which are randomly selected data points, that will be used to map the original data to a lower-dimensional space. This greatly reduces the computational cost of spectral clustering, making it more suitable for larger datasets.

This brings us to the main question we will address in this research:

"To what extent can the computational efficiency of Spectral Clustering be enhanced by using Extremely Randomized trees to calculate the similarity matrix and by implementing the Locally Linear Landmarks approach?"

2 Theoretical Background

In Section 2.1 we will first get acquainted with the general spectral clustering problem, which will lay the foundation for the rest of the research. Subsequently, in Section 2.2 we will explain our used similarity measure *RatioRF* and the type of decision tree we used to construct a random forest. Then, in Section 2.3 the Locally Linear Landmark technique will be explained. Section 2.4 will conclude this chapter with a description of the Adjusted Rand Index and the purity measure, which are the measures that are used to quantify the performance of spectral clustering.

2.1 General spectral clustering problem

There are multiple formulations of the spectral clustering problem, which vary in factors like the type of similarity measure that is used, or the kind of normalization applied to the graph Laplacian, which we will consider in more detail later. We will first explain the unnormalized definition of the problem, which we can then later use to build upon.

The goal of spectral clustering is to partition data into clusters, where data points that are within the same cluster are more similar to each other than to data points in other clusters. As the name implies, spectral clustering does this by exploiting the spectral properties of the data. This involves analyzing the eigenvalues and eigenvectors of matrices that are derived from the data.

2.1.1 Input variables

Given a dataset $Y_{N\times d} = (y_1, y_2, ..., y_N)$ consisting of N data points, there are two main parameters that need to be determined before the spectral clustering algorithm can be applied: the similarity matrix $W \in \mathbb{R}^{N\times N}$ and the number of clusters k.

The similarity matrix $W \in \mathbb{R}^{N\times N}$ is a symmetric matrix that contains all the pairwise similarities $s_{ij} \geq 0$ between the data points. The construction of the matrix depends greatly on the type of similarity measure that is chosen. Similarity measures that are often used in spectral clustering are the Gaussian Kernel Function, the k-nearest Neighbors or the Euclidean distance. We chose to use the recently proposed similarity measure *RatioRF* [2], which is a random forest-based measure that appears to yield great results. In Section 2.2.1 we will consider this measure in more detail.

The number of clusters, k, that ought to be found in the dataset should also be specified beforehand. There exist multiple methods to determine this value, like the elbow method [15] or using the gap statistic [16]. Within this research we assume knowledge of the number of clusters. In particular we will only use datasets that are available on the UCI ML Repository [17], which usually states how many clusters the dataset can be divided in.

2.1.2 Unnormalized minimization problem

As stated before, the goal of spectral clustering is to find the best way to split the data into clusters such that the similarity between clusters is minimized and the similarity within clusters is maximized. We can reformulate the spectral clustering problem as a graph partitioning problem by stating that we want to find a partition of the similarity graph W that minimizes the weights of the edges that connect the different clusters and maximizes

the weights of the edges within a cluster, where nodes $i, j \in W$ are only connected by an edge if there weights are larger than zero, $w_{i,j} > 0$. Focusing only on this first part, that is minimizing the weights of the edges between distinct clusters, we achieve this by using the unnormalized Laplacian $L = D - W$, where matrix $D \in \mathbb{R}^{N \times N}$ is the degree matrix, which is a diagonal matrix with on the diagonal the degrees of each node of a graph. A graph Laplacian is a matrix that is defined in such a way that it contains properties of a graph that can be very useful in graph partitioning problems.

This graph partitioning problem is encapsulated in the following optimization problem, which will be fully derived in Section 2.1.3:

$$\underset{X \in \mathbb{R}^{N \times k}}{min} \quad Tr(X^T L X), \quad s.t. \quad X^T X = I \tag{1}$$

Here, the goal is to find an orthogonal matrix $X \in \mathbb{R}^{N \times k}$ that minimizes the trace of the matrix $X^T L X$, where $L = D - W$ is the unnormalized Laplacian. The trace of a matrix is the sum of its diagonal elements.

This is a standard form of a trace minimization problem, where we can use the Rayleigh-Ritz theorem to solve this problem. This theorem states that in the context of spectral clustering and Laplacian matrices the solution to this problem is given by the k smallest eigenvalues of L, which contain important information about the connectivity and thus the clusters of the graph. The solution matrix X is real-valued, however to find a concrete partition of the dataset we need to convert it to a discrete solution. We can acquire this by using the k-means algorithm [7] on the rows of X.

The complete mathematical derivation of unnormalized spectral clustering will be done in the following Section 2.1.3. A reader only interested in the methodology, may skip to Section 2.1.4, where the basics of normalized spectral clustering are presented.

2.1.3 Mathematical foundation of spectral clustering

Given a dataset $Y_{N \times k} = (y_1, ..., y_N)$ and an associated similarity matrix $W \in \mathbb{R}^{N \times N}$, we can construct a similarity graph G with vertex set $V = (v_1, ..., v_N)$, where vertices are only connected by an edge if it has a nonzero weight $w_{i,j} > 0$. We want to find a partition of the graph that minimizes the weight of the edges between different clusters. We will define the weight between two clusters $A, B \subset V$ as $W(A, B) = \sum_{i \in A, j \in B} w_{i,j}$ and the complement of a set $A \subset V$ as $\overline{A} = \{v_i \in V | v_i \notin A\}$.

Clustering as a graph partitioning problem

We can define the spectral clustering problem as a graph partitioning problem: for a given $k \in \mathbb{Z}_+$, find a partition $A_1, ..., A_k$ that minimizes $\text{Cut}(A_1, ..., A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i)$. This was first introduced by Stoer and Wagner in [14], but it does not always lead to satisfactory results because there is no constraint on the cluster size. This often results in one single vertex being separated from the rest. With cluster size, denoted by $|A|$, the amount of vertices in a set is meant. To deal with this constriction, we can use the RatioCut objective function, introduced in [4], which makes sure that the sets $A_1, ..., A_k$

are not too small. It is defined in the following way:

$$\text{RatioCut}(A_1, ..., A_k) \overset{def}{=} \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{\text{Cut}(A_i, \overline{A}_i)}{|A_i|}$$

Notice that the RatioCut takes on a smaller value if the clusters A_i are larger. This way it tries to balance the clusters in terms of number of vertices in each set.

Solution for 2 clusters

To simplify the situation, we will first set $k = 2$ and solve the associated minimization problem:

$$\underset{A \subset V}{min} \quad \text{RatioCut}(A, \overline{A}) \tag{2}$$

There is a very useful property of the unnormalized Laplacian L that we can use to reformulate the minimization problem in a more convenient way. The proof of this property can be found in Appendix 8.1, and the proposition is formulated as follows:

Proposition: *For every vector* $f \in \mathbb{R}^N$ *we have* $\quad f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$.

We now introduce an indicator vector $f = (f_1, ..., f_N)' \in \mathbb{R}^N$ with entries

$$f_i = \begin{cases} \sqrt{|\overline{A}|/|A|} & \text{if} \quad v_i \in A \\ \sqrt{|A|/|\overline{A}|} & \text{if} \quad v_i \in \overline{A}. \end{cases} \tag{3}$$

Combining the proposition with our newly defined indicator vector f, we arrive at the following equation. The complete derivation can be found in Appendix 8.1.

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

Now we can rewrite our minimization problem of Equation (2) as follows:

$$\underset{A \subset V}{min} \quad \text{RatioCut}(A, \overline{A}) \quad = \quad \underset{A \subset V}{min} \quad \frac{f'Lf}{|V|} \quad = \quad \underset{A \subset V}{min} \quad \frac{f'Lf}{N} \tag{5}$$

Since N is a constant, we can remove it from the problem without changing the nature of the problem. Now we will look at two properties that vector f has that we can incorporate in the final version of our minimization problem. First, we see that f is orthogonal to the constant vector $\mathbf{1}_N = (1, 1, ..., 1)_{1\times N}$:

$$f \cdot \mathbf{1} = \sum_{i=1}^{N} f_i = \sum_{i \in A} \sqrt{\frac{|\overline{A}|}{|A|}} - \sum_{i \in \overline{A}} \sqrt{\frac{|A|}{|\overline{A}|}} = |A|\sqrt{\frac{|\overline{A}|}{|A|}} - |\overline{A}|\sqrt{\frac{|A|}{|\overline{A}|}}$$

Taking the square of the right-hand side shows that it equals 0. We can thus conclude that f is orthogonal to the constant one vector.

Lastly, we will look at the squared norm of f and see that it is equal to N:

$$||f||^2 = \sum_{i=1}^{n} f_i^2 = \sum_{i \in A} \frac{|\overline{A}|}{|A|} + \sum_{i \in \overline{A}} \frac{|A|}{|\overline{A}|} = |A|\frac{|\overline{A}|}{|A|} + |\overline{A}|\frac{|A|}{|\overline{A}|} = |\overline{A}| + |A| = |V| = N$$

Combining the properties that hold for our defined f, we can write the minimization problem that we defined in Equation (5) as:

$$\min_{A \subset V} \quad f'Lf \quad s.t. \quad f \perp \mathbf{1}, \quad f_i \text{ defined as in (3)}, \quad ||f|| = \sqrt{N}$$

We have now formulated it as a discrete optimization problem, since the entries of f can only be two particular values, Equation (3). We can relax it and transform it to a continuous problem by allowing f_i to take arbitrary values in $\mathbb{R}$. This relaxation is justified, since it makes the problem computationally more efficient and easier to implement, while it still remains a good approximation to the original discrete problem. The relaxed and final optimization problem of unnormalized spectral clustering with $k = 2$ is given below:

$$\min_{f \in \mathbb{R}^N} \quad f'Lf \quad s.t. \quad f \perp \mathbf{1}, \quad ||f|| = \sqrt{N}$$

By applying the Rayleigh-Ritz theorem to this problem, we can find the solution vector f. This theorem states that in the context of spectral clustering and Laplacian matrices the solution to this problem is given by the second smallest eigenvalue of L, which contains important information about the connectivity and thus the clusters of the graph. This yields a real-valued solution vector f, but we want a discrete partition of a graph which means we have to transform it back to a discrete indicator vector. In spectral clustering this is often done by the k-means algorithm [7], where we first consider the coordinates f_i as points in $\mathbb{R}$ and cluster them accordingly into two clusters $C, \overline{C}$. Then we finalize the clustering by the following heuristic:

$$\begin{cases} v_i \in A & \text{if} \quad f_i \in C \\ v_i \in \overline{A} & \text{if} \quad f_i \in \overline{C} \end{cases}$$

We have now obtained a solution to the unnormalized spectral clustering problem, for $k = 2$. We will now generalize this for arbitrary k, while falling back on the calculations we already made for $k = 2$.

Solution for arbitrary k

For an arbitrary number of clusters k, the goal of spectral clustering is to find a partition of the vertex set V into k sets $A_1, ..., A_k$ that minimizes the weights between distinct clusters:

$$\min_{A_1,...,A_k \subset V} \quad \text{RatioCut}(A_1, ..., A_k) \tag{6}$$

Now, instead of introducing a single indicator vector f, we will define k indicator vectors $h_j = (h_{1,j}, ..., h_{N,j})'$ by

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_j|} & \text{if} \quad v_i \in A_j \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, ..., N; j = 1, ..., k) \tag{7}$$

Then, we construct a matrix $X \in \mathbb{R}^{N \times k}$ by setting the k indicator vectors as its columns: $H = (h_1, ..., h_k)$. Since each vertex v_i can only be in one subset A_j, the dot product of

two distinct indicator vectors will always be equal to zero. This means that all the h_j are orthogonal to each other. We also show that the squared norm of each indicator vector equals one:

$$||h_j||^2 = \sum_{j \in A_j} \frac{1}{|A_j|} = |A_j| \cdot \frac{1}{|A_j|} = 1$$

Since we proved that all indicator vectors h_j are orthogonal to each other and all have unit length, we can conclude that all h_j are orthonormal to each other. This implies that matrix X is also orthonormal, meaning $X^T X = I$. Similar to the calculations for the case that $k = 2$, we can derive that

$$h_i' L h_i = \frac{\text{cut}(A_i, \overline{A}_i)}{|A_i|}. \tag{8}$$

Focusing on the left side of Equation (8), using the structure of X we can state that it is equal to the $(i,i)th$ element of $X^T LX$: $h_i' L h_i = (X^T LX)_{ii}$. Combining these two findings, we can reformulate the RatioCut as follows:

$$\text{RatioCut}(A_1, ..., A_k) \overset{def}{=} \sum_{i=1}^{k} \frac{\text{Cut}(A_i, \overline{A}_i)}{|A_i|} = \sum_{i=1}^{k} h_i' L h_i = \sum_{i=1}^{k} (X^T LX)_{ii} = Tr(X^T LX)$$

The original minimization problem of Equation (6), can be rewritten as a trace minimization problem.

$$\underset{A_1,...,A_k}{min} \text{Tr}(X^T LX) \quad s.t. \quad X^T X = I, \quad X \text{ defined as in (7)}$$

Now, similar to the relaxation of the case where $k = 2$, we transform this discrete minimization problem to a continuous problem by allowing matrix X to take real values. We now ended up at the standard form of an unnormalized spectral clustering problem for arbitrary k:

$$\underset{X \in \mathbb{R}^{N \times d}}{min} \text{Tr}(X^T LX) \quad s.t. \quad X^T X = I$$

To find the solution matrix X, we apply the same approach as for $k = 2$. The Rayleigh-Ritz theorem tells us to set the smallest k eigenvectors of L as columns for matrix X. Then we apply the k-means algorithm to the rows of X and we end up with the discrete partition of graph G.

This concludes the complete mathematical background of unnormalized spectral clustering.

2.1.4 Normalized minimization problem

Now that we have found matrix X using the unnormalized Laplacian, we already have a fair approximation to the solution of the spectral clustering problem. However, this solution only focuses on minimizing the weights between distinct clusters, while concentrating on maximizing the weights within clusters can be a good addition to the optimization problem. This can be achieved by using a normalized Laplacian. These are more robust to variations

in graph structures and vertex degrees. The two most popular normalized Laplacians are the symmetric Laplacian L_{sym} [9] and the random walk Laplacian L_{rw} [12]. We have implemented both Laplacians, but we will only state the altered optimization problem with L_{rw} here.
The random walk normalized Laplacian is defined in the following way:

$$L_{rw} = D^{-1}L$$

Using this new definition, the optimization problem of Equation (1) can be reformulated to the following, where $L = D - W$:

$$\begin{aligned} \min_{X \in \mathbb{R}^{N \times d}} \quad & Tr(X^T L X) \\ s.t. \quad & X^T D X = I \end{aligned} \tag{9}$$

The derivation of this can be found in [6].

The strategy to finding the solution matrix $X \in \mathbb{R}^{N \times k}$ is similar to the approach explained in Section 2.1.2, with as only difference that X should consist of the first k eigenvectors of L_{rw} as columns, instead of L.

2.1.5 Complexity of spectral clustering

One of the biggest bottlenecks of spectral clustering is its complexity. While the technique consists of multiple computational steps, such as constructing the graph Laplacian L and clustering, the eigendecomposition of the graph Laplacian is the most computationally intensive step. This involves finding the eigenvalues and corresponding eigenvectors of the Laplacian. For a dense $N \times N$ matrix, the eigendecomposition typically has a complexity of $O(N^3)$ [3], which dominates the complexity of spectral clustering. When dealing with large datasets, this can get expensive really fast.

2.2 Random forest-based similarity measure

2.2.1 Similarity measure RatioRF

As similarity measure we chose the relatively newly proposed measure *RatioRF* [2] for calculating the similarity matrix. We will explain this measure using an example. Imagine a decision tree T, where each node represents a binary test θ that either results in a 'yes' (left) or a 'no' (right). To obtain a similarity value for data points x and y, we will send both objects to traverse through the same decision tree, tracking both paths they take. If x and y (would) have the same result on a certain test in the tree, we color the node green. Similarly, we color a node red if the data points do not agree on a test. See Figure 1 for a decision tree where data points x and y have travelled through. Now we will define the decision tree similarity measure *RatioDT* as the ratio of the number of tests in the paths on which they agree to the total number of tests encountered by both data points. The higher the value of $RatioDT(x, y)$, the higher the similarity between data points x and y. The mathematical formulation is stated below:

$$RatioDT(x, y) = \frac{|X \dot{\cap} Y|}{|X \dot{\cap} Y| + |X \dot{-} Y| + |Y \dot{-} X|}$$

X and Y represent the feature sets of data points x and y, respectively. These sets consist

of the test results obtained along the path from the root of tree T to the leaf node where the respective data point resides. $|X \dot{\cap} Y|$ is the set of test results on which x and y agree, among the test results in the union of their feature sets. $|X \dot{-} Y|$ is the set of test results of x that y disagrees with.

Going back to Figure 1, we see that x and y agree on 5 tests, also counting the tests that only one of them came across, so $|X \dot{\cap} Y| = 5$. Data points x and y each disagree with each other in two tests along their respective paths, making $|X \dot{-} Y| = |Y \dot{-} X| = 2$. Using this information, we can thus conclude that $RatioDT = \frac{5}{5+2+2} = \frac{5}{9}$. This example showcases the decision tree similarity measure $RatioDT$ based on a single tree T. Now imagine we have a random forest, consisting of multiple decision trees $T_1, ..., T_n$. We can easily extend $RatioDT$ to a random forest-based measure $RatioRF$ by averaging over the n decision trees:

$$RatioRF(x, y) = \frac{1}{n} \sum_{t=1}^{n} RatioDT_t(x, y)$$

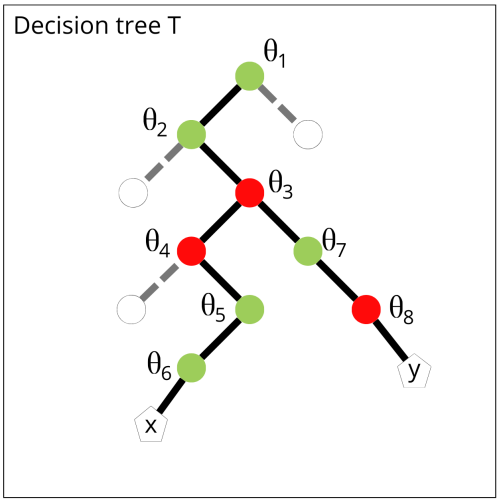

FIGURE 1: Paths of data points x, y in decision tree T with binary tests θ_i

A more detailed explanation of the similarity measures $RatioDT$ and $RatioRF$ can be found in [2].

2.2.2 Type of decision tree used in random forest

To calculate the similarity measure $RatioRF$, we need a trained random forest consisting of trained decision trees. The paper that introduced this similarity measure [2] used binary trees to make a random forest. While binary trees are a common option for this, we propose the utilization of extremely randomized trees. We will shortly state the main difference between the two and highlight the advantages of extremely randomizes trees. For a more elaborate explanation of random forests and decision trees, we refer to [13].

The main difference between the two types of decision trees lies in the used split criterion. Binary trees try to find the optimal split for every node. It achieves this by searching through all features and their possible thresholds and selecting the best feature and threshold, based on a certain metric that quantifies the homogeneity of clusters within the resulting splits. Commonly used metrics are information gain and the Gini criterion. While this leads to accurate models, it is also computationally expensive since it does an intensive search at each node. Extremely randomized trees do not search for the optimal split. They introduce additional randomness by selecting features and thresholds randomly. This reduces overfitting, because in the ensemble of a random forest it implies more diverse and less correlated trees. Using extremely randomized trees also reduces the computationally efficiency greatly, which makes it more suitable for dense large datasets.

2.3 Spectral clustering using the Locally Linear Landmark approach

As we saw in Section 2.1.5, the computational efficiency of spectral clustering is limited when dealing with large datasets. One way to reduce this complexity is to convert the original optimization problem to a reduced spectral clustering problem, so that we are dealing with smaller matrices. A technique that achieves this is the Locally Linear Landmarks (LLL) approach, first introduced by M. Vladymyrov and M. À . Carreira-Perpiñàn in [19] for manifold learning.

2.3.1 Reduced spectral clustering problem

Before we dive into LLL, we shortly look back at the original spectral clustering problem with the normalized Laplacian L_{rw}. The optimization problem we are trying to solve on dataset $Y_{N\times d} = (y_1, ..., y_N)$ was as follows:

$$\begin{aligned} \min_{X\in\mathbb{R}^{N\times k}} \quad & Tr(X^T L X) \\ s.t. \quad & X^T D X = I \end{aligned}$$

LLL aims to reformulate this problem to a reduced problem, meaning that the eigendecomposition in the process will be done on smaller matrices such that it is computationally more efficient.

It achieves this by introducing landmarks $\tilde{Y}_{L\times d} = (\tilde{y}_1, ..., \tilde{y}_L) \subset Y$. Landmarks are randomly selected data points that will be used to map the original data to a lower-dimensional space. Each data point y_i will be represented as a linear combination of the K_Z nearest landmarks. To be able to make this projection, we first need to construct the projection matrix $Z \in \mathbb{R}^{L\times N}$. This matrix has as its rows the $L << N$ landmarks $\tilde{Y}$ and as its columns the N data points Y. Its elements are the weights that approximate each data point as a linear combination of the K_Z nearest landmarks. This means that each column has exactly K_Z nonzero elements. This linear mapping can be described by the following relation:

$$Y \approx \tilde{Y} Z \tag{10}$$

We have chosen to use the Euclidean distance as metric to find the closest landmarks, since it is computationally efficient. To construct matrix Z, we need to solve the following minimization problem:

$$\min_{Z} \quad ||Y - \tilde{Y}Z||^2, \quad s.t. \quad 1^T Z = 1^T$$

The objection function $||Y - \tilde{Y}Z||$ measures the difference between the original data and the mapped data, which we want to minimize. The constraint ensures that the columns of the projection matrix Z sum to one, which results in Z becoming translation invariant. This is a standard minimization problem that can be solved using linear least squares.

After constructing projection matrix Z, we can use it to project our data points Y to a reduced dimensional space Y_{red}, using $Y_{red} = YZ$.
The most important assumption of LLL is that the dependence between landmarks and

data points in the high-dimensional space, given by Equation (10), is also preserved in the low-dimensional space:

$$X \approx \tilde{X} Z \tag{11}$$

If we substitute this into the original spectral clustering problem we derived in Section 2.1.4, we get the following reduced spectral clustering problem:

2.3.2 Complexity of spectral clustering using LLL

The main goal of applying the Locally Linear Landmark approach is to reduce the complexity of spectral clustering. As we saw in Section 2.1.5, spectral clustering usually has a complexity of $O(N^3)$ when dealing with dense $N \times N$ matrices. This was mainly because of the intensive eigendecomposition of the $N \times N$ Laplacian. The LLL technique reduces this step to an eigendecomposition of an $L \times L$ matrix, where we assume that $L << N$. This already reduces the computational cost drastically. However, there are additional computations performed during the LLL approach that do not occur in the original process. Matrices Z, $\tilde{A}$ and $\tilde{B}$ need to be constructed and the final projection done by Equation (11) also adds to the computational cost. The construction of projection matrix Z is computationally dominated by the construction of the N pointwise Gram matrices G. We will not go into detail on this matrix, but an explanation can be found in [19]. Because of the linear relationship between the datapoints and landmarks, G is sparse and it has only K_Z nonzero elements in each row. This leads to a complexity for the construction of the projection matrix Z of $O(NdK_Z^2)$. Each row of matrix G The cost of computing $\tilde{A}$ and $\tilde{B}$ is $O(K_Z N^2)$, since there are $(N \times N = N^2)$ elements in the resulting matrix and each element required $O(K_Z)$ operations. The final projection has a cost of $O(N \times L \times d)$, because it results in a matrix $X_{N \times d}$, where each element required $O(L)$ operations. Combining these results with the $O(L^3)$ complexity due to the eigendecomposition, we end up with a total complexity of $O(K_Z N^2 + Nd(L + K_Z^2) + L^3)$ for spectral clustering with the LLL approach. This is asymptotically much faster than the single eigendecomposition $O(N^3)$ for general spectral clustering, since we assumed $L << N$.

2.4 Performance measures

We used two different performance measures to evaluate our clustering results of our spectral clustering implementation, namely the Adjusted Rand Index (ARI) [5] and the purity measure [8]. We chose these, because they were also used in the *RatioRF* paper [2] which makes it suitable to compare results.

2.4.1 Adjusted Rand Index

The Adjusted Rand Index is a measure that quantifies the similarity between two distinct clusterings. We will use it to evaluate the results between our found clustering and the true labeling, which we assume to be given. It is a correction of the Rand Index (RI), which is a basic similarity measure between two clusterings. The difference between them is that the ARI takes into account that there could also occur some agreement between two clusterings by chance. So as the name implies, the Adjusted Rand Index adjusts the Rand Index to also consider this possibility. Its calculation is as follows:

Given is a dataset consisting of N data points, and two distinct clusterings C_1, C_2 of the dataset, namely the predicted clustering and the true labeling, respectively. We will first determine the Rand Index by evaluating the clustering of all pairs of data points. Each pair falls into one of the four following possibilities:

1. True Positive (TP): pairs of data points that are in the same cluster in both the predicted clustering C_1 and the true clustering C_2.
2. True Negative (TN): pairs of data points that are in different clusters in both the predicted clustering C_1 and the true clustering C_2.
3. False Positive (FP): pairs of data points that are in the same cluster in the predicted C_1, but in different clusters in C_2.
4. False Negative (FN): pairs of data points that are in different clusters in C_1, while they are in the same clusters in the true clustering C_2..

Then the Rand Index is defined as the ratio of the total number of agreements to the total number of pairs of datapoints:

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

To take the possibility of random agreement into account, we will calculate the expected value of the Rand Index, $E[RI]$:

$$E[RI] = \frac{TP \cdot (TP + FP) + TN \cdot (FN + TN)}{(TP + FP + FN + TN)^2}$$

Now we have everything we need to calculate the Adjusted Rand Index:

$$ARI = \frac{RI - E[RI]}{1 - E[RI]}$$

The ARI value ranges from -1 to 1, where 1 means the two clusterings C_1 and C_2 completely agree, 0 means it agrees the same as done by chance, and -1 states that the clusterings completely disagree, so it is even worse than random.

2.4.2 Purity measure

The purity measure assesses how well data points that truly belong to the same cluster are actually grouped in the same cluster. We assume we have a found clustering C_1 and a true labeling C_2. To calculate the purity measure we look at each found cluster in C_1 individually.

Within each cluster we find the true label that is most frequent in that cluster, so this is the label that most data points in the cluster actually belong to. We assign this label to the cluster. Then the purity measure is the proportion of data points assigned to the correct label.

A schematic overview of this situation is given in Figure 2. The colours represent the three distinct clusters that were found in the process of clustering 14 data points. The objects in the clusters are the data points, where its shape reflects their true label. The clusters are assigned to the label that is most frequent in the cluster, visualised by the line going out of the cluster. Now we count all the correctly assigned data points, which for the red cluster equals 4, for blue 3 and for green also 3. This totals to 10 correctly assigned data points, which means the purity value of this clustering is equal to $\frac{10}{14} = 0.71$.
The purity value ranges from 0 to 1, with 1 indicating that the clustering has been performed perfectly.

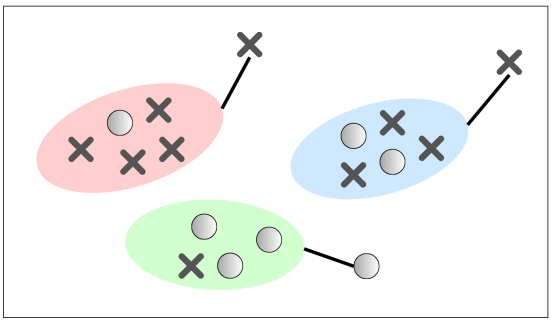

FIGURE 2: Demonstration of the purity measure

3 Methods

3.1 Python implementation

The paper that introduced the *RatioRF* similarity measure [2] published the Matlab files [1] they used for their experiments online. These files include the training of the random forest to calculate the similarity matrix using *RatioRF*, the unnormalized and normalized version of spectral clustering and the Adjusted Rand Index. We downloaded these and converted all of them to Python, which was an intensive job since all files are integrated with each other. This choice was made, because Python is open-source and we have a greater expertise in Python compared to Matlab. We added the purity measure and implemented the Locally Linear Landmarks approach from scratch. Our Python files can be found on `https://gist.github.com/zoevanh`.

3.2 Datasets

To test our implementations we used 9 different datasets, which are all available on the UCI ML Repository [17]. These datasets are often used to test classification methods. They differ in number of objects, number of features and number of clusters. An overview of the datasets is given in Table 1.

The datasets all have different formats and structures, so we had to perform some data preparation steps for each dataset. This consisted mainly of splitting the data points from their true labels and deleting data points that contained missing values. Since spectral clustering and our use of random forest are unsupervised techniques, the column containing all labels had to be separated. These labels will be used in the end to evaluate our clustering results.

TABLE 1: Details of the datasets that we used in testing

Name	#objects	#features	#clusters
Iris	150	4	3
Wine	178	13	3
Glass	214	9	4
WBC	683	9	2
BTissue	106	9	6
Heart	297	13	2
Lung	32	54	3
Parkinsons	195	22	2
Pima	768	8	2

3.3 Experimental parameters

In Section 4 we will present our found results. Since we are working with Python files that were mainly converted from Matlab files associated with [2], we will first attempt to reproduce their spectral clustering results using binary trees as close as possible. This will confirm that the code we are using is at least producing similar results, which makes it more reliable to build upon further. Then we will generate results using extremely randomized trees in stead of binary trees. Besides the mentioned performance measures, we will also look at the runtime of the algorithms. This way we can observe whether the complexity indeed reduced. Lastly, we will present our results with the Locally Linear Landmarks approach. In the overall process there are numerous parameters that need to be determined beforehand. We will go through them in the following sections.

3.3.1 Spectral clustering results using binary trees

Our first goal is to reproduce the results presented in [2] as close as possible, since we used their code as a starting point for our implementation. This implies that we will mimic their simulation in terms of parameters.

To train their random forest they used binary trees, which are built by randomly selecting 80 percent of the training set data. They experimented with different forest sizes, namely 50, 100 and 200, and the proportion of features to allow during the selection of the optimal split, 0.5 or 1. This results in 6 different forest parametrizations. For the selection of the best split the classical Gini criterion was used. They repeated every experiment 30 times on 12 datasets, using 4 different spectral clustering algorithms. This leads to averaged results over 720 ($6 \times 30 \times 4$) experiments for each dataset.

Unfortunately we can not perform all these experiments for each dataset, as we have time restrictions. We decided to settle with 1 forest parametrization, building 50 binary trees and using half of the features during the optimal split selection. This enhances the randomization inside each tree. We also built the trees using 80 percent of the training

set data and we employed the Gini criterion. We repeated each experiment 30 times, but only with one spectral clustering method that uses the normalized symmetric Laplacian [9]. This eventually comes out to 30 ($1 \times 30 \times 1$) experiments for each of our datasets, for which the results will be aggregated to averaged accuracies.

3.3.2 Spectral clustering results using extremely randomized trees

After using binary trees to build our random forests, we will transfer to the use of extremely randomized trees. We will again use 1 forest parametrization, building 50 trees and using half of the features during the random split selection. The trees are built with all the data points, instead of 80 percent as for the binary trees. We repeat each experiment 30 times using only the normalized symmetric Laplacian method for spectral clustering. This comes out to 30 ($1 \times 30 \times 1$) experiments for each of our datasets, for which the results will be aggregated to averaged accuracies.

3.3.3 Spectral clustering results with the LLL approach

To generate our results of spectral clustering with the Linear Landmarks Approach we will keep the parameters that are needed to train the extremely randomized forest the same as presented in Section 3.3.2.

The LLL approach does also include some choices in parameters, namely the number of landmarks L used to map the data to a lower-dimensional space and the number of landmarks K_Z used to build a linear reconstruction of each data point.

The main goal of Locally Linear Landmarks revolves around the choice of the number of landmarks L. The idea is to make L as small as possible, especially keeping it significantly smaller than the total number of datapoints N. Conversely, increasing the number of landmarks increases the accuracy of the approximation. The choice of L determines the dimension of our mapped data points, which directly impacts the matrix on which the eigendecomposition will be performed. Due to time constraints, we did not do an optimization on the number of landmarks for each dataset. Instead, we settled for a fixed value of L equal to 50 percent of the number of datapoints, $L = 0.5 \cdot N$. In a related study [18] experiments were repeated 5 times varying the numbre of landmarks, logarithmically spaced from $L = 8$ to $L = N - 10$. Given the considerable range, we deemed our approach of setting $L = 0.3 \cdot N$ to be sufficient for our research.

Each data point is expressed as a linear combination of the nearest K_Z landmarks. Following the approach in [19] we decided to keep K_Z consistent for all the data points. Due to time restrictions, we did not perform an optimization for K_Z. However, in the study of [19] they concluded that in the context of manifold learning the value of K_Z should fall within the range of $[\hat{d}+1, d+1]$, where $\hat{d}$ is the intrinsic local dimensionality of the manifold and d is the number of features. Although our task differs from manifold learning, we decided to go for a value of $K_Z = 0.8 \cdot d$, which aligns with this interval. Since this approach was the only one found in the literature, we decided it would be sufficient enough for our research.

4 Results

4.1 Averaged results using binary trees

The results of spectral clustering where binary trees are used to calculate the similarity matrix, are presented in Table 2 in terms of the Adjusted Rand Index and the purity measure. The second and fourth columns contain the results that were presented in the paper that introduced the similarity measure *RatioRF* [2]. The third and fifth columns contain the results of our implementation. This makes it simple to compare results. In Section 3.3.1 the parameters that are used to generate these results can be found. The results are averaged over 720 experiments for the results of [2] and averaged over 30 experiments for our results.

TABLE 2: Spectral clustering results using binary trees, averaged over multiple experiments

	ARI		Purity	
Name	*RatioRF* paper [2]	Our research	*RatioRF* paper [2]	Our research
Iris	0.707	0.721	0.878	0.889
Wine	0.743	0.806	0.899	0.933
Glass	0.175	0.205	0.558	0.600
WBC	0.883	0.892	0.970	0.973
BTissue	0.379	0.373	0.597	0.597
Heart	0.252	0.274	0.748	0.762
Lung	0.113	0.096	0.530	0.527
Parkinsons	0.154	0.121	0.754	0.754
Pima	0.066	0.057	0.659	0.657

4.2 Averaged results using extremely randomized trees

In Table 3 and Table 4, we present our main spectral clustering results. Table 3 compares the results between the use of binary trees and extremely randomized trees for calculating the similarity matrix, in terms of the Adjusted Rand Index and the purity measure. Table 4 presents the comparison of the runtime of the spectral clustering algorithm between the use of the different decision trees. In both tables the results are averaged over 30 experiments. The used parameters can be found in Section 3.3.2.

4.3 Averaged results with the Locally Linear Landmarks approach

Table 5 and Table 6 cover the spectral clustering results with the Locally Linear Landmarks approach implemented. We will compare it with the results of our general spectral clustering method, which were already presented in the previous section. Table 5 shows the spectral clustering results in terms of the Adjusted Rand Index and the purity measure.

TABLE 3: Averaged spectral clustering results using binary trees or extremely randomized trees

	ARI		Purity	
Name	Binary	ExtTrees	Binary	ExtTrees
Iris	0.721	0.684	0.888	0.871
Wine	0.806	0.844	0.933	0.948
Glass	0.205	0.178	0.600	0.581
WBC	0.892	0.817	0.973	0.952
BTissue	0.373	0.385	0.597	0.620
Heart	0.274	0.405	0.762	0.819
Lung	0.096	0.184	0.527	0.582
Parkinsons	0.121	0.152	0.754	0.754
Pima	0.057	0.054	0.657	0.651

TABLE 4: Averaged runtime of the spectral clustering algorithm using binary trees or extremely randomized trees

	Time in seconds	
Name	Binary	Ext
Iris	14.46	3.35
Wine	101.72	4.45
Glass	87.87	5.74
WBC	64.32	47.71
BTissue	49.20	2.15
Heart	29.32	10.65
Lung	2.98	0.68
Parkinsons	133.89	5.06
Pima	120.66	101.73

Table 6 gives the comparison of the runtime of the algorithm with or without the LLL approach implemented. Both tables contain averaged results over 30 experiments. The parameters that were used to generate these results are presented in Section 3.3.3.

TABLE 5: Averaged spectral clustering results with or without the Locally Linear Landmarks approach

	ARI		Purity	
Name	General SC	LLL SC	General SC	LLL SC
Iris	0.684	0.505	0.871	0.667
Wine	0.844	0.555	0.948	0.775
Glass	0.178	0.0160	0.581	0.381
WBC	0.817	0.239	0.952	0.743
BTissue	0.385	0.157	0.620	0.388
Heart	0.405	0.256	0.819	0.729
Lung	0.184	0.138	0.582	0.556
Parkinsons	0.152	-0.009	0.754	0.752
Pima	0.054	0.004	0.651	0.653

TABLE 6: Averaged runtime of the spectral clustering algorithm with or without the Locally Linear Landmarks approach

	Time in seconds	
Name	General SC	LLL SC
Iris	3.35	1.22
Wine	4.45	1.58
Glass	5.74	2.10
WBC	47.71	19.57
BTissue	2.15	0.79
Heart	10.65	3.88
Lung	0.68	0.30
Parkinsons	5.06	2.07
Pima	101.73	42.64

5 Discussion and recommendations

In this section we will explain the results that were presented in Section 4. We will also go into the limitations of our research and propose recommendations for further research.

5.1 Analysis of our results

Our first goal was to attempt to reproduce the results of [2], that used binary trees for calculating the similarity matrix, as close as possible with our Python implementation. This would indicate whether our implementation is working as expected, setting their results as a basis, before expanding our research to explore more techniques with spectral clustering. In Table 2 our results are represented next to their results. It can be easily observed that our results closely resemble those presented in [2], despite the fact that our results are averaged over just 30 experiments, compared to their 720 experiments. The biggest difference in results in terms of the Adjusted Rand Index and the purity measure is 0.063 and 0.042, respectively, and these are both in favor of our research. From now on we will thus assume that our implementation of the general spectral clustering algorithm with *RatioRF* as similarity measure is performing effectively and is reliable.

Next, we tried to replace binary trees with extremely randomized trees to construct the similarity matrix. The results can be found in Section 4.2. We expected a significant reduction in runtime, since it introduces more randomness and it avoids the need for an exhaustive search to find the optimal split. Since this additional randomness also benefits overfitting and increases robustness to noisy data, we expected no serious reduction in performance. Our results confirm our expectations of both the runtime and performance. Table 4 shows that for all the 9 datasets the computational cost significantly reduced in terms of runtime when extremely randomized trees were used. In Table 3 it can be seen that for only 4 datasets there is a slight reduction in performance, with 0.075 and 0.021 being the largest differences in ARI and purity, respectively. For 5 datasets the runtime reduced with the use of extremely randomized trees, while its performance increased. And for 3 of these datasets, the runtime was reduced by a factor of ~ 23. This brings us to our conclusion that the use of extremely randomized trees is a very good alternative when used for calculating similarity measures based on decision trees in the context of spectral clustering.

Lastly we implemented the Locally Linear Landmarks [19] approach. Since it is a dimensionality reduction technique, it is expected that the runtime of the algorithm would reduce considerably. As the approach yields an approximate solution to the spectral clustering problem, it should not be unexpected that there will be some trade-off happening in terms of performance. Our results with the use of the LLL approach are presented in 4.3. Again, our results mostly confirm our expectations. First looking at the runtimes, which can be found in Table 6, it can be observed that for all the datasets the runtime significantly decreased with a factor of more than 2. In Table 5 the performance of spectral clustering with the LLL approach is presented. There is indeed a reduction in performance visible when the LLL method is employed. Especially the Adjusted Rand Indices experienced a fair reduction. Dataset *WBC*, which had an ARI of 0.817 with the general algorithm, now has an ARI equal to 0.239, which is significantly worse. The *Parkinsons* dataset, which although it already had a poor ARI, now has a negative ARI, indicating that the clustering is even worse than by random chance. The purity measure also reflects a reduction in performance, but considerably less. Our results for the ARI are not as expected unfortunately, which indicates potential limitations or inefficiencies in our LLL implementation. These will be elaborated in the following section.

5.2 Limitations and recommendations

There are numerous important factors in the LLL approach that could have a significant impact on the performance and runtime if not carefully considered. For instance, both the number of landmarks L and the number of landmarks K_Z used to express data points as a linear combination play crucial roles in the LLL approach. If L and K_Z are too small, there may be situations where there are not enough landmarks close to express a data point as a local linear combination. However, the computational cost grows as L and K_Z get bigger. This emphasizes the importance of carefully selecting these parameters. We decided to settle with fixed values, both based on literature sources that did not completely align in objective. It could be that we settled for too little K_Z. Taking these perspectives into account, the optimization of these variables would definitely be an interesting topic for further research.

Another aspect of the LLL approach that deserves some more consideration is the is the selection of landmarks location. We decided to just select L landmarks randomly, which enhances the randomness in the algorithm and keeps the computational cost low. However, with this approach situations can arise where the landmarks are non-uniformly distributed, resulting in a poor representation of the higher-dimensional dataset. In [19] an interesting alternative method that attempts to make the landmark location as close to uniform as possible is proposed. This method involves randomly selecting $L + M$ landmarks, finding M pairs of closest landmarks and subsequently discarding one landmark from each pair. Another straightforward validation step is to evaluate the location of landmarks after they are randomly selected. This can be done by numerous techniques, such as visual inspection, analyzing the standard deviation of the pairwise distances or examining the density of data points surrounding each landmarks. If these evaluations indicate a poor representation of the original dataset, one could consider revisiting the landmarks selection step.

In our research we used the euclidean distance as a metric to calculate the K_Z nearest landmarks for each data point. Euclidean distance is effective in capturing linear relations and its computational cost is low. However, it is sensitive to scale and outliers. Exploring alternative distance metrics that take the specific characteristics of the data into account could be an interesting topic for further research.

The fact that there is such a big difference in the ARI and the purity measure for the LLL approach results, is not that uncommon. The ARI considers all pairs of data points and their cluster labels, including outliers and noisy data, while the purity measure only considers the majority class within each cluster. This makes the ARI less robust to outliers, which can have an impact on the performance. Some of our used datasets, like Parkinsons and WBC have an imbalanced class distribution, meaning that one class dominates the others. This will always lead to better purity results. This same theory is valid for the situation where there are only 2 clusters, which was the case for 4 out of 9 of our datasets. In these cases the purity measure may not provide a thorough assessment of clustering quality, as it tends to favor clusters with dominant class labels. Further research may consider exploring more performance measures to evaluate the clustering results. For instance, the Silhouette Score [11] and the Davies-Bouldin Index [10] would be good alternatives to explore, because they are both more suited for datasets with a small number of clusters.

A general limitation of our research is the limited number of experiments conducted to derive our averaged results, due to our time constraint. For each dataset we executed 30

experiments, using only a single forest parametrization and 1 spectral clustering method. To get more accurate and reliable results, exploring more forest parametrizations would be a good starting point for further research. Finally, our experiments were only conducted on small to moderate datasets. A valuable addition to extend our research would be to conduct experiments on significantly larger datasets to evaluate how our implementation would perform under different scales.

6 Conclusion

To conclude this research, let us briefly come back to our research question:

"To what extent can the computational efficiency of Spectral Clustering be enhanced by using Extremely Randomized trees to calculate the similarity matrix and by implementing the Locally Linear Landmarks approach?"

Focusing first on the use of extremely randomized trees instead of binary trees to calculate the similarity matrix, we have shown that this significantly reduces the computational cost of spectral clustering in terms of runtime, while still maintaining a good performance. All datasets that we used in our experiments show a serious reduction in runtime, with only 4 out of 9 datasets showing a slight reduction in performance. For 5 datasets the runtime reduced, while its performance increased in terms of the Adjusted Rand Index and the purity. For 3 of these datasets, the runtime was reduced by a factor of ~ 23. We can thus conclude that extremely randomized trees are a good alternative when using random forest to calculate pairwise similarities in the context of spectral clustering.

As for our implementation of the Locally Linear Landmarks approach, we definitely showed its potential as a dimensionality reduction technique in terms of runtime. While it demonstrated a reduction in runtime by a minimum of 50% in all datasets, our results also revealed a notable trade-off in terms of performance. Especially for most of the Adjusted Rand Indices there was a significant decrease visible. Our purity results were decent, considering the fact that the solution obtained through the LLL approach is approximate. We still advocate for the LLL approach as a good alternative for dimensional reduction techniques, but we recommend further research efforts towards optimizing its parameters.

A result worth mentioning, although not directly tied to our research question, is our successful Python implementation of the general spectral clustering algorithm. This implementation incorporates a random forest for calculating the similarity matrix and offers the option to use the Linear Locally Landmarks approach. Since Python is an open-source platform, our implementation can easily be built upon to explore more of spectral clustering. This contribution facilitates further research and advancements in the field of spectral clustering, growing our understanding and potential applications of this algorithm.

7 Bibliography

References

[1] Manuele Bicego. *Code and Datasets for Manuele Bicego's Publications*. http://profs.sci.univr.it/~bicego/code.html. Accessed: <insert date accessed>. 2023.

[2] Manuele Bicego, Ferdinando Cicalese, and Antonella Mensi. "RatioRF: a novel measure for Random Forest clustering based on the Tversky's Ratio model". In: *IEEE Transactions on Knowledge and Data Engineering* (2021). My implementation of RF, pp. 1–1. ISSN: 1041-4347. DOI: 10.1109/TKDE.2021.3086147.

[3] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. 4th. Johns Hopkins University Press, 2013. ISBN: 978-1421407944.

[4] L. Hagen and A.B. Kahng. "New spectral methods for ratio cut partitioning and clustering". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11.9 (1992), pp. 1074–1085. DOI: 10.1109/43.159993.

[5] Lawrence Hubert and Phipps Arabie. "Comparing partitions". In: *Journal of Classification* 2 (1 Dec. 1985), pp. 193–218. ISSN: 0176-4268. DOI: 10.1007/BF01908075.

[6] Ulrike von Luxburg. "A Tutorial on Spectral Clustering". In: (Nov. 2007).

[7] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: 1967. URL: https://api.semanticscholar.org/CorpusID:6278891.

[8] C. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. 2008.

[9] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On spectral clustering: analysis and an algorithm". In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. NIPS'01. Vancouver, British Columbia, Canada: MIT Press, 2001, pp. 849–856. DOI: 10.5555/2980539.2980649.

[10] Frédéric Ros, Rabia Riad, and Serge Guillaume. "PDBI: A partitioning Davies-Bouldin index for clustering evaluation". In: *Neurocomputing* 528 (2023), pp. 178–199. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.01.043. URL: https://www.sciencedirect.com/science/article/pii/S0925231223000528.

[11] Ketan Rajshekhar Shahapure and Charles Nicholas. "Cluster Quality Analysis Using Silhouette Score". In: *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. 2020, pp. 747–748. DOI: 10.1109/DSAA49011.2020.00096.

[12] Jianbo Shi and J. Malik. "Normalized cuts and image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (8 2000), pp. 888–905. ISSN: 01628828. DOI: 10.1109/34.868688.

[13] YY Song and Y Lu. "Decision tree methods: applications for classification and prediction". In: *Shanghai Archives of Psychiatry* 27.2 (2015), pp. 130–135. DOI: 10.11919/j.issn.1002-0829.215044.

[14] Mechthild Stoer and Frank Wagner. "A simple min-cut algorithm". In: *J. ACM* 44.4 (July 1997), pp. 585–591. ISSN: 0004-5411. DOI: 10.1145/263867.263872. URL: https://doi.org/10.1145/263867.263872.

[15] Robert L. Thorndike. "Who belongs in the family?" In: *Psychometrika* 18 (4 Dec. 1953), pp. 267–276. ISSN: 0033-3123. DOI: 10.1007/BF02289263.

[16] Robert Tibshirani, Guenther Walther, and Trevor Hastie. "Estimating the Number of Clusters in a Data Set Via the Gap Statistic". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 63.2 (Jan. 2002), pp. 411–423. ISSN: 1369-7412. DOI: 10.1111/1467-9868.00293. eprint: https://academic.oup.com/jrsssb/article-pdf/63/2/411/49590410/jrsssb_63_2_411.pdf. URL: https://doi.org/10.1111/1467-9868.00293.

[17] UCI Machine Learning Repository. *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml. Accessed: <Access Date>.

[18] Max Vladymyrov and Miguel Á. Carreira-Perpiñán. "Fast, accurate spectral clustering using locally linear landmarks". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 3870–3879. DOI: 10.1109/IJCNN.2017.7966344.

[19] Max Vladymyrov and Miguel Á. Carreira-Perpiñán. "Locally Linear Landmarks for Large-Scale Manifold Learning". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Hendrik Blockeel et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 256–271. ISBN: 978-3-642-40994-3. DOI: 10.1007/978-3-642-40994-3_17.

8 Appendices

8.1 Mathematics behind general spectral clustering

Proof of proposition of L

For every vector $f \in \mathbb{R}^N$ *we have* $\quad f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$.

Proof

$$\begin{aligned} f'Lf &= f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{i=1}^{n} d_i f_i^2\right) \\ &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2. \qquad \blacksquare \end{aligned} \tag{12}$$

Derivation of relation between proposition of L and RatioCut

We will first need to derive the following statements before we can do the whole derivation of Equation 4:

$$\begin{aligned} \mathrm{Cut}(A, \overline{A}) &\overset{def}{=} \frac{1}{2}\sum_{i=1}^{2} W(A_i, \overline{A}_i) \\ &= \frac{1}{2}W(A_1, \overline{A}_1) + \frac{1}{2}W(A_2, \overline{A}_2) \\ &= \frac{1}{2}W(A_1, A_2) + \frac{1}{2}W(A_2, A_1) \\ &= W(A_1, A_2) = W(A, \overline{A}) \\ &= \sum_{i \in A, j \in \overline{A}} w_{i,j} \end{aligned} \tag{13}$$

$$\begin{aligned} \mathrm{RatioCut}(A, \overline{A}) &\overset{def}{=} \sum_{i=1}^{2} \frac{\mathrm{Cut}(A_i, \overline{A}_i)}{|A_i|} \\ &= \frac{\mathrm{Cut}(A_1, \overline{A}_1)}{|A_1|} + \frac{\mathrm{Cut}(A_2, \overline{A}_2)}{|A_2|} \\ &= \frac{\mathrm{Cut}(A, \overline{A})}{|A|} + \frac{\mathrm{Cut}(\overline{A}, A)}{|\overline{A}|} \\ &= \mathrm{Cut}(A, \overline{A})\left(\frac{1}{|A|} + \frac{1}{|\overline{A}|}\right) \end{aligned} \tag{14}$$

Using Equation (13) and Equation (14), we can do the following derivation:

$$
\begin{aligned}
f'Lf &\overset{Eq.12}{=} \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 \\
&= \frac{1}{2}\sum_{i\in A, j\in\overline{A}} w_{ij}\left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}}\right)^2 + \frac{1}{2}\sum_{i\in\overline{A}, j\in A} w_{ij}\left(-\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}}\right)^2 \\
&\overset{Eq.13}{=} \frac{1}{2}\mathrm{Cut}(A,\overline{A})\left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}}\right)^2 + \frac{1}{2}\mathrm{Cut}(\overline{A},A)\left(-\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}}\right)^2 \\
&= \frac{1}{2}\mathrm{Cut}(A,\overline{A})\left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2\right) + \frac{1}{2}\mathrm{Cut}(\overline{A},A)\left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2\right) \\
&= \mathrm{Cut}(A,\overline{A})\left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2\right) \\
&= \mathrm{Cut}(A,\overline{A})\left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + \frac{|\overline{A}|}{|\overline{A}|} + \frac{|A|}{|A|}\right) \\
&= \mathrm{Cut}(A,\overline{A})\left(\frac{|A|+|\overline{A}|}{|A|} + \frac{|A|+|\overline{A}|}{|\overline{A}|}\right) = \mathrm{Cut}(A,\overline{A})\left(\frac{|V|}{|A|} + \frac{|V|}{|\overline{A}|}\right) \\
&= |V|\cdot\mathrm{Cut}(A,\overline{A})\left(\frac{1}{|A|} + \frac{1}{|\overline{A}|}\right) \overset{Eq.14}{=} |V|\cdot\mathrm{RatioCut}(A,\overline{A})
\end{aligned}
$$

8.2 Derivation of general spectral clustering to LLL problem

$$
\begin{aligned}
&\min_{X\in\mathbb{R}^{N\times d}} \quad Tr(XLX^T), \quad s.t. \quad XDX^T = I \\
&\qquad\qquad \downarrow X = \tilde{X}Z \\
&\min_{\tilde{X}\in\mathbb{R}^{L\times d}} \quad Tr(\tilde{X}ZL(\tilde{X}Z)^T), \quad s.t. \quad \tilde{X}ZD(\tilde{X}Z)^T = I \\
&\qquad\qquad \downarrow \\
&\min_{\tilde{X}\in\mathbb{R}^{L\times d}} \quad Tr(\tilde{X}(ZLZ^T)\tilde{X}^T), \quad s.t. \quad \tilde{X}(ZDZ^T)\tilde{X}^T = I \\
&\qquad\qquad \downarrow \\
&\min_{\tilde{X}\in\mathbb{R}^{L\times d}} \quad Tr(\tilde{X}\tilde{A}\tilde{X}^T), \quad s.t. \quad \tilde{X}\tilde{B}\tilde{X}^T = I
\end{aligned}
$$

with $L \times L$ reduced similarity matrices $\tilde{A} = ZLZ^T$ and $\tilde{B} = ZDZ^T$.