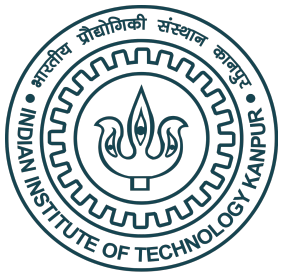
Spectral Clustering: Theory and Applications*

Submitted by:

Arkajyoti Bhattacharjee †‡

Supervised by:

Dr. Amit Mitra †



Submitted on:

22^{th} April, 2022

*This report has been prepared towards the partial fulfillment of the requirements of the course *MTH552A: Statistical & AI Techniques in Data Mining.*

†Department of Mathematics & Statistics, Indian Institute of Kanpur, India.

‡201277, M.Sc. Statistics (Final year).

Abstract

In this report, we present a class of popular clustering algorithms called Spectral Clustering algorithms. We introduce graph theoretic notations required to understand the report. We discuss similarity graphs and graph Laplacians, along with their important properties. Three popular clustering algorithms are presented. Choice of optimal number of clusters, similarity functions, similarity graphs and graph Laplacians are also discussed. We then present Spectral clustering through different looking glasses. Finally, we apply Spectral clustering to simulated and real life datasets. This report is primarily based on Von Luxburg (2007).

Contents

1 Introduction

One of the most widely used class of unsupervised machine learning algorithms is that of clustering algorithms. During the early stages of every data analysis, we first try to observe the data and try to figure out inherent similarities in the data. Spectral clustering (Von Luxburg (2007)) is one of the most popular and widely used clustering algorithms and shows an edge against common clustering algorithms like the "k-means" algorithm (Forgy (1965), Hastie et al. (2009)).

In this report, we attempt to provide the basics of spectral clustering from the ground-up and apply it to synthetic as well as real life data. The rest of the report is organized as follows. We introduce graph theoretic notations required to understand the report in Section (2). We discuss similarity graphs and graph Laplacians, along with their important properties in Section (3) and (4). Three popular clustering algorithms are presented in Section (5). Choice of optimal number of clusters and graph Laplacians are also discussed in Sections (6), (7). We then present Spectral clustering through different looking glasses in Section (8). Finally, we apply Spectral clustering to simulated and real life datasets in Section (9).

2 Graph Theory: Some definitions and notations

The objective of clustering is to group data having similar or homogeneous attributes together and separate out data that are heterogeneous or dissimilar (in some respect) into different groups. Let's say we have a set of n data points $X_1, \ldots, X_n$ and there exists some similarity measure $s_{ij} \geq 0$ for all $i, j = 1(1)n$. Based on only this information, we can represent the data in the form of a graph or rather, a *similarity graph*. Let it be denoted by $G = (V, E)$, where V is the set of all vertices and the E is the set of all edges of the graph. Each data point X_i is represented by a vertex v_i and two vertices are connected by an edge e_{ij} if the vertices v_i and v_j have a corresponding similarity measure s_{ij} above a particular threshold; the edge e_{ij} is weighted by the similarity measure s_{ij}, where more weight means the associated vertices are more similar.

2.1 Clustering problem formulation based on graphs

We want to obtain a partition of the graph such that the edges between similar objects have higher weights and edges between dissimilar objects have lower weights.

This motivates us to study some basics of graph theory that will be required in spectral clustering.

2.2 Notations

Let $G = (V, E)$ be a weighted, undirected graph, where $V = (v_1, \ldots, v_n)$ is the set of vertices. We assume the weights $w_{ij} \geq 0$, where w_{ij} is the weight carried by the edge connecting the vertices v_i and v_j. We define the assocaited weighted *adjacency matrix*

$$W = ((w_{ij}))_{i,j=1(1)n}.$$

The graph being unweighted simply implies that W is a symmetric matrix. We define the degree associated with a vertex v_i with

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

The degree matrix, is then, defined to be

$$D = diag(d_1, \ldots, d_n).$$

For $\phi \neq A, B \subset V$, we define

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

We will be using two measures of "size" of $A \subset V$:

$$|A| \stackrel{def}{=} \text{cardinality of the set A},$$

$$vol(A) \stackrel{def}{=} \sum_{i \in A} d_i.$$

We say that $A \subset V$ is *connected* if any two vertices in A can be joined by a path such that all intermediate points also lie in A; it is called a *connected component* if it is connected and if there are no connections between vertices in A and its complement. The nonempty sets $A_1, \ldots, A_k$ form a partition of the graph $G = (V, E)$ if $A_i \cap A_j = \phi$ and $\cup_{i=1}^{k} A_i = V$.

3 Similarity Graphs

There are many similarity graphs used in the literature. Here, we mention three popular similarity graphs:

***k*-nearest neighbor graphs:** Here, we connect v_i with v_j if v_j is amongst the

k-nearest neighbors of v_i, "nearest" being w.r.t. some measure of distance d_{ij}. Observe that although v_j may be amongst the k-nearest neighbors of v_i, it may not be so the other way round. So, the graph becomes directed. To make it undirected, one of the following are usually done: (1) ignore the directions, i.e., connect v_i with v_j if either of the two are amongst the k-nearest neighbors of each other. The graph obtained is called the *k-nearest neighbor graph*; (2) connect v_i with v_j if both are amongst the k-nearest neighbors each other. The graph obtained is called the *mutual k-nearest graph*.

ε-neighborhood graph: We connect those points for which the pairwise distances are smaller than the radius ε.

Fully connected graph: We connect those points which share a positive similarity with each other and weigh the connected edges with s_{ij}. Similarity functions, like the Gaussian function similarity function

$$s(x_i, x_j) = \frac{1}{\sqrt{2\pi}} e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}, \tag{1}$$

are used to model the local neighborhoods. The σ in the Gaussian similarity function controls the width of the neighborhoods.

3.1 Construction

Based on the similarity graphs defined in Section (3), we look into the construction of those similarity graphs.

3.1.1 Choice of similarity function

The first step of constructing a similarity graph is choosing a similarity function. Although the choice depends on the domain of the data, we need to make sure that the objects deemed "similar" by the similarity function are indeed meaningful. For example, if a similarity function indicates two text documents to be very similar, we need to verify whether they even belong to the same category. In the common case, where the data lives in the Euclidean space $\mathbb{R}^d$, a popular choice for the similarity function is the Gaussian similarity function defined in (1).

3.1.2 Choice of similarity graph

There does not exist any universal choice of a similarity graph from those mention in Section (3). Each have their separate pros and cons.

In the ε-neighborhood graph, choosing the radius ε is non-trivial and is often difficult, particularly if the data where the distances between points are different in different regions of the space (Von Luxburg (2007)).

However, the k-nearest graph can connect data on "different scales". The k-nearest neighbor graph also has the property of breaking into several disconnected components if there are high density regions which are considerably far off. The resulting similarity matrix is sparse.

The mutual k-nearest neighbor graph tends to connect points within regions of constant density, but does not connect regions of different densities with each other. So, it can be considered lying midway between" the ε-neighborhood graph and the k-nearest neighbor graph. The mutual k-nearest neighbor graph, thus, seems particularly well-suited if we want to detect clusters of different densities.

The Gaussian similarity function in (1) is often associated with the fully connected graph. This is because it provides more weights to local points and positive but lower weights to distant points. The resulting similarity matrix is, however, not sparse.

Von Luxburg (2007) recommends choosing the k-nearest neighbor graph as the first choice because of its simplicity, resultant sparse adjacency matrix W and is robust to the choice of similarity graph parameters.

3.1.3 Choice of similarity graph parameters

After choosing the similarity graph, one has to choose the connectivity parameter k, or ε. This is a non-trivial task. Although many asymptotics results exist, they do not help us dealing with finite sample.

For some rules of thumb, we suggest the interested reader to see Von Luxburg (2007) for a comprehensive review.

4 Graph Laplacian: Different types and their properties

In this section, we will assume that the graph G is weighted and undirected, with weight matrix W. Unless otherwise stated, the eigenvectors are unnormalized. The eigenvalues are considered in an increasing order of magnitude.

For detailed proofs of the propositions, we refer the reader to Von Luxburg (2007).

4.1 The unnormalized graph Laplacian

The unnormalized graph Laplacian matrix L is defined as:

$$L = D - W.$$

See Mohar et al. (1991), Mohar (1997) for more details. We summarize the important properties of L relevant to spectral clustering in the following proposition.

Proposition 1 (**Properties of L**, Von Luxburg (2007))**.** *The matrix L satisfies the following properties:*
1. For every vector $f \in \mathbb{R}^n$, we have

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

2. L is symmetric and positive semi-definite.
3. The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$.
4. L has n non-negative, real-valued eigenvalues $0 \leq \lambda_1 \leq \cdots \leq \lambda_n$.

The following proposition gives an important property of L useful in spectral clustering.

Proposition 2 (**Number of connected components and the spectrum of L**, Von Luxburg (2007))**.** *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.*

4.2 The normalized graph Laplacian

In the literature, there are two matrices which are called normalized graph Laplacians. They are:

$$L_{sym} \stackrel{def}{=} D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2};$$

$$L_{rw} \stackrel{def}{=} D^{-1}L = I - D^{-1}W.$$

Here, L_{sym} is a symmetric matrix and L_{rw} is closely related to a random walk. The following proposition summarizes some important properties of these matrices.

Proposition 3 (**Properties of L_{sym} and L_{rw}**, Von Luxburg (2007))**.** *The normalized Laplacians satisfy the following properties:*
1. For every $f \in \mathbb{R}^n$ we have

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 .$$

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 solve the generalized eigen-problem $Lu = \lambda Du$.
4. 0 is an eigenvalue of L_{rw} with the constant one vector $\mathbb{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbb{1}$.
5. L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 \leq \lambda_1 \leq \cdots \leq \lambda_n$.

The following proposition relates the multiplicity of the eigenvalue 0 of the normalized graph Laplacian to the number of connected components.

Proposition 4 (**Number of connected components and spectra of L_{sym} and L_{rw}**, Von Luxburg (2007))**.** *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of both L_{rw} and L_{sym} equals the number of connected components $A_1, \ldots, A_k$ in the graph. For L_{rw}, the eigenspace of 0 is spanned by the indicator vectors $\mathbb{1}_{A_i}$ of those components. For L_{sym}, the eigenspace of 0 is spanned by the vectors $D^{1/2}\mathbb{1}_{A_i}$.*

5 Three Spectral Clustering Algorithms

In this section we provide three most common spectral clustering algorithms. We assume we have n data points $x_1, \ldots, x_n$ which can be arbitrary objects. We measure their pairwise similarities $s_{ij} = s(x_i, x_j)$ by some similarity function which is symmetric and non-negative, and denote the corresponding similarity matrix by $S = (s_{ij})_{i,j=1,\ldots,n}$.

Algorithm 1 Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
Construct a similarity graph by one of the ways described in Section (3). Let W be its weighted adjacency matrix.
Compute the unnormalized Laplacian L.
Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.
Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
for $i = 1, \ldots, n$ **do**
    let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of U.
Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | y_j \in C_i\}$.

There are two different versions of normalized spectral clustering, depending on the type of the normalized graph Laplacian used.

Algorithm 2 Normalized spectral clustering, Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
Construct a similarity graph by one of the ways described in Section (3). Let W be its weighted adjacency matrix.
Compute the unnormalized Laplacian L.
Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenvalue problem $Lu = \lambda Du$.
Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
for $i = 1, \ldots, n$ **do**
    let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of U.
Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | y_j \in C_i\}$.

Observe that this algorithm uses the generalized eigenvectors of L, which according to Proposition (3) correspond to the eigenvectors of the matrix L_{rw}. So in fact, the algorithm works with eigenvectors of the normalized Laplacian L_{rw}, and hence, called normalized spectral clustering.

Algorithm 3 Normalized spectral clustering, Ng et al. (2001)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
Construct a similarity graph by one of the ways described in Section (3). Let W be its weighted adjacency matrix.
Compute the normalized Laplacian L_{sym}.
Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{sym}.
Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
Form the matrix $T \in \mathbb{R}^{n \times k}$ from U by normalizing the rows to norm 1, that is set $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$.
for $i = 1, \ldots, n$ **do**
    let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of T.
Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | y_j \in C_i\}$.

In all the above algorithms, the main trick is to change the representation of the points x_i to points $y_i \in \mathbb{R}^k$. Owing to the properties of the graph Laplacians, this change of representation is useful as it enhances the cluster-properties in the data, so that clusters can be trivially detected in the new representation by the k-means algorithm.

6 Choice of the cluster number

The problem of choosing the optimal cluster number in present in almost every clustering algorithm. For the case of spectral clustering, we focus on choosing k based on the eigenmap heuristic, which can be used for all the three graph Laplacians. The goal is to choose k such that the eignevalues $\lambda_1, \ldots, \lambda_k$ are very small and λ_{k+1} is relatively large. Although, the eigenmap heuristic usually works well when the data exhibits evident clusters, in ambiguosu cases it yields ambiguous results (Von Luxburg (2007)).

7 Choice of the graph Laplacian

Of the three graph Laplacians mentioned in Section (4), the one which should be used for computing the eigenvectors is not known apriori and there does not exist any universal choice. However, there are strong reasons for choosing normalized graph Laplacians over unnormalized graph Laplacians and furthermore, choosing L_{rw} over L_{sym}.

7.1 Clustering Objectives

Consider the $k = 2$ case for ease of understanding. The main objectives of clustering are two-fold:
1. We want to minimize the between-cluster similarity. In the graph setting, this means to minimize $cut(A, \bar{A})$.
2. We want to maximize the within-luster similarity. In the graph setting, this means to maximize $W(A, A)$ and $W(\bar{A}, \bar{A})$.

Both RatioCut and NCut take care of the first point by explicitly incorporating $cut(A, \bar{A})$ in the objective function.

For the second point, note that

$$W(A, A) = W(A, V) - W(A, \bar{A}) = vol(A) - cut(A, \bar{A}).$$

So, the within-cluster similarity is maximized when $vol(A)$ is large and $cut(A, \bar{A})$. This is exactly what we achieve by minimizing NCut and thus, NCut criterion takes care of the second objective as well.

For the RatioCut, observe that the objective is to maximize $|A|$ and $|\bar{A}|$ instead of $vol(A)$ and $vol(\bar{A})$. Now, $|A|$ and $|\bar{A}|$ are not necessarily related to the within-cluster similarity since it depends on the edges and not on the number fo vertices in A.

In short, NCut takes care of both of the fundamental objectives of clustering, whereas RatioCut takes care of only the first objective.

7.2 Asymptotics

We assume that the data points $X_1, \ldots, X_n$ are a random sample from some underlying distribution P. We are interested to see if the results of spectral clustering converge to some useful partition of the underlying feature space $\mathscr{X}$ as $n \to \infty$.

For both the normalized spectral clustering algorithms, itcan be proved that L_{sym} converges almost surely to an operator U on the space of continuous functions on $\mathscr{X}, \mathscr{C}(\mathscr{X})$. This convergence implies that the eigenvalues and eigenvectors of L_{sym} converge to those of U. This in turn can be transformed to a statement about the convergence of normalized spectral clustering. It can be shown that the partition which is induced on $\mathscr{X}$ by the eigenvectors of U can be interpreted similar to the random walks interpretation of spectral clustering. Under very mild conditions,

usually satisfied in real world applications, all consistency statements about normalized spectral clustering hold, for both L_{sym} and L_{rw} (Von Luxburg (2007)).

For the unnormalized spectral clustering, it can be proved that it may fail converge, or that it can converge to trivial solutions which construct single-point clusters of the data space. It can be proved that the matrix $(1/n)L$ itself converges to some limit operator T on $\mathscr{C}(\mathscr{X})$ as $n \to \infty$, the spectral properties of this limit operator T can be as bad as to prevent the convergence of spectral clustering. It is possible to construct examples that poor results are possible even in finite sample cases.

So, form a statistical point of view, it is better to use normalized spectral clustering over unnormalized spectral clustering.

Finally, observe that the eigenvectors of L_{rw} are cluster indicator vectors $\mathbb{1}_{A_i}$, while the eigenvectors of L_{sym} are additionally multiplied with $D^{1/2}$, which might cause numerical inaccuracies computationally. All the reasons above, thus, recommend using L_{rw}.

8 Spectral Clustering: Different Points of View

In this section, we look at Sepctral Clustering through different looking glasses.

8.1 Graph Cut Point of View

Given a similarity graph with adjacency matrix W, the direct and simplest way to construct a graph partition is to solve the mincut problem. For a given number k of subsets, the mincut approach simply consists in choosing a partition $A_1, \ldots, A_k$ which minimizes

$$cut(A_1, \ldots, A_k) \stackrel{def}{=} \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A}_i)$$

. For the $k = 2$ case, the mincut is relatively straightforward (see Stoer and Wagner (1997)). However, practically it may ked to unsatisfactory outcomes. Often the problem is that the solution of mincut separates one individual vertex from the rest of the graph. One way around this problem is to explicitly request that the sets $A_1, \ldots, A_k$ are "reasonably large". The two most popular objective functions to encode this are RatioCut (Hagen and Kahng (1992)) and the normalized cut Ncut (Shi and Malik (2000)), which we define below:

$$RatioCut(A_1, \ldots, A_k) \stackrel{def}{=} \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|},$$

$$NCut(A_1,\ldots,A_k) \stackrel{def}{=} \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i,\bar{A}_i)}{vol(A_i)} = \sum_{i=1}^{k}\frac{cut(A_i,\bar{A}_i)}{vol(A_i)}.$$

Here, the objective functions aim for the clusters to be "balanced". Unfortunately, introducing balancing conditions makes the previously simple to solve mincut problem become NP hard (Wagner and Wagner (1993)). Spectral clustering is a way to solve relaxed versions of those problems; relaxing Ncut leads to normalized spectral clustering, while relaxing RatioCut leads to unnormalized spectral clustering.

8.1.1 Approximating RatioCut for $k = 2$

Here, our goal is to solve the optimization problem

$$\min_{A\subset V} RatioCut(A,\bar{A}).$$

For a given $A \subset V$, we define the vector $f = (f_1,\ldots,f_n)' \in \mathbb{R}^n$ with entries

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & v_i \in A \\ -\sqrt{|\bar{A}|/|A|} & v_i \in \bar{A}. \end{cases}$$

Now, it can be shown that

$$f'Lf = |V|RatioCut(A,\bar{A}),$$

along with $f'\mathbb{1} = 0$ and $||f||^2 = n$.

So, the optimization problem can be restated as:

$$min_{A\subset V} f'Lf \text{subject to} f \perp \mathbb{1}, ||f|| = \sqrt{n}.$$

Since f_is can take only two values - 0 and 1 - this is a discrete optimization problem. A natural relaxation is to allow f_i to take values in $\mathbb{R}$. The relaxed problem can be solved by the Rayleigh-Ritz theorem, by which f is the eigenvector corresponding to the second smallest eigenvalue of L. To return from the continuous optimization problem to the disxcrete optimization problem, what most spectral clustering algorithms do is consider the coordinates $fi \in R$ and cluster them into two groups C and $\bar{C}$ by the k-means clustering algorithm; then choose

$$\begin{cases} v_i \in A & f_i \in C \\ v_i \in \bar{A} & f_i \in \bar{C}. \end{cases}$$

This is exactly what unnormalized spectral clustering does for $k = 2$.

8.2 Approximating RatioCut for arbitrary k

Following the procedure for approximating RatioCut for $k = 2$, we can extend it similarly for an arbitrary k. Given a partition of V into k sets $A_1, \ldots, A_k$, we define k indicator vectors $h_j = (h_{1,j}, \ldots, h_{n,j})'$ by

$$h_{ij} = \begin{cases} 1/\sqrt{|A_j|} & v_i \in A \\ 0 & v_i \in \bar{A}. \end{cases}$$

We then define the matrix $H \in \mathbb{R}^{n \times k}$ as

$$H = [h_1 : \cdots : h_k].$$

Clearly, the columns of H are orthogonal. See that

$$h_i' L h_i = \frac{cut(A_i, \bar{A}_i)}{|A_i|} = (H'LH)_{ii}$$

.
So, we get:

$$RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} h_i' L h_i = \sum_{i=1}^{k} (H'LH)_{ii} = trace(H'LH).$$

Thus, the problem of minimizing RatioCut can be translated to:

$$\min_{A_1, \ldots, A_k} trace(H'LH) \text{subject to} H'H = I.$$

Similar to the $k = 2$ case, the above optimization can be relaxed to

$$\min_{H \in \mathbb{R}^{n \times k}} trace(H'LH) \text{subject to} H'H = I.$$

A version of the Rayleigh-Ritz theorem indicates that the solution to the above problem is the matrix containing the firsts k eigenvectors of L as columns. The solution matrix H is in fact the matrix U in the unnormalized clustering algorithm (1). To convert the real solution to a discretized solution, we again perform k-menas clustering on the rows of U, leading to the general unnormalized clustering algorithm in Section (5).

8.3 Approximating NCut

Following a similar approach to that of RatioCut, it can be shown that on relaxing the minimization of NCut, we get normalized spectral clustering.

8.3.1 $k = 2$ case

We define the cluster indicator vector f as:

$$f_i = \begin{cases} \sqrt{vol(\bar{A})/vol(A)} & v_i \in A \\ -\sqrt{vol(A)/vol(\bar{A})} & v_i \in \bar{A}. \end{cases}$$

Similar to the RatioCut case, we observe that $(Df)'\mathbb{1} = 0, f'Df = vol(V)$, and $f'Lf = vol(V)NCut(A,\bar{A})$.

So, the minimization problem of NCut can be translated to:

$$\min_A f'Lf \text{subject to} Df \perp \mathbb{1}, f'Df = vol(V).$$

On relaxing $f \in \mathbb{R}^n$ and substituting $g \stackrel{def}{=} D^{1/2}f$, the problem reduces to:

$$\min_{g \in \mathbb{R}^n} g'D^{-1/2}LD^{-1/2}g \text{subject to} g \perp /D^{-1/2}\mathbb{1}, ||g||^2 = vol(V).$$

Note that $D^{-1/2}LD^{-1/2} = L_{sym}$, $D^{-1/2}\mathbb{1}$ is the first eigenvector of L_{sym}, and $vol(V)$ is constant. So again based on Rayleigh-Ritz theorem,we get the solution of the above optimization problem as the second eigenvector of L_{sym}. Putting back $f = D^{-1/2}g$ and using Proposition (3), we see that f is the second eigenvector of L_{rw}.

8.3.2 $k > 2$ case

We define the indicator vector $h_j = (h_{1,j}, \ldots, h_{n,j})'; j = 1, \ldots, k$, by:

$$h_{ij} = \begin{cases} 1/\sqrt{vol(A_j)} & v_i \in A \\ 0 & v_i \in \bar{A}. \end{cases}$$

We define $H = [h_1 :, \ldots, : h_k]$. Note that the columns of H are orthogonal, $h_i'Dh_i = cut(A_i, barA_i)/vol(A_i)$.

So, the optimization problem of NCut can be translated to

$$\min_{A_1,\ldots,A_k} trace(H'LH) \text{subject to } H'DH = I.$$

On relaxing the discreteness and putting $T = D^{1/2}H$ educes the above to :

$$\min T \in \mathbb{R}^{n \times k} trace(T'D^{-1/2}LD^{-1/2}T) \text{subject to } T'T = I.$$

Following similar steps to that of RatioCut, we obtain the final solution of H consists of the first k eigenvectors of the matrix L_{rw}, or the first k eigenvectors of the generalized eigenvalue problem $Lu = \lambda Du$. Thus, this relaxation yields in the normalized spectral clustering algorithm of Shi and Malik (2000).

8.4 Random Walk Point of View

In this section, we show how spectral clustering can be viewed as a random walk on a graph that stays within a cluster for long and seldom jumps between clusters.

The transition probability, p_{ij}, of jumping from vertex v_i to vertex v_j is given by:

$$p_{ij} \stackrel{def}{=} w_{ij}/d_i$$

. This menas that the transition probability matrix, P of the random walk is given by:

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

The random walk always possess a unique stationary distribution $\pi = (\pi_1, \ldots, \pi_n)'$ with $\pi = di/vol(v)$, provided the graph is non-bipartite and connected.

Observe that $L_{rw} = I - P$. So, if λ is an eigenvalue of L_{rw} with eigenvector u if and only 1-λ is an eigenvalue of P with eigenvector u.

The following proposition provides an equivalence between NCut and the transition probabilities of the random walk.

Proposition 5 (**NCut via transition probabilities**, Meilă and Shi (2001))**.** *Let G be connected and non bipartite. Assume that we run the random walk $(X_t)_{t\in\mathbb{N}}$ starting with $X+0$ in the stationary distribution π. For disjoint subsets $A, B \subset V$, denote by $P(B|A) := P(X_1 \in B | X_0 \in A)$. Then:*

$$Ncut(A, \bar{A}) = P(A|\bar{A}) + P(\bar{A}|A).$$

The above proposition can be interpreted as follows: when minimizing Ncut, we actually look for a cut through the graph such that a random walk seldom transitions from $\bar{A}$ to A and vice versa.
Another link between graph Laplacians and random walks can be made using the commute distance on a graph. We refer the interested reader to see Von Luxburg (2007) for more details.

9 Applications

In this section, we look into some applications of Spectral clustering algorithms.

9.1 Synthetic Data

9.1.1 k-means vs. Spectral Clustering

Here, we compare k-means clustering with Spectral clustering on two different toy datasets. We can clearly see that spectral clustering performs better on both the datasets as its clustering is more "meaningful".

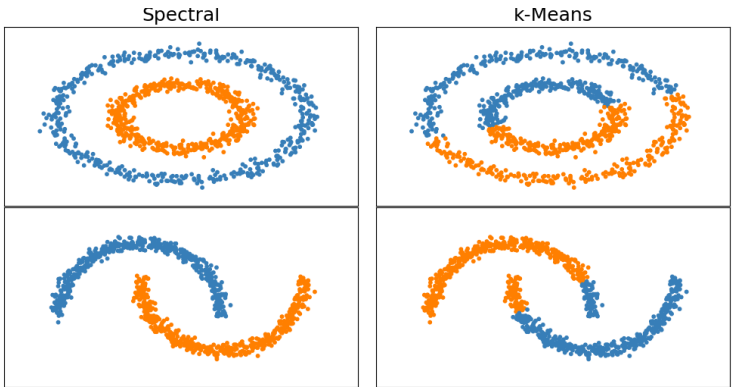


Figure 1: Comparison between Spectral and k-means clustering on two toy datasets.

9.1.2 Image segmentation

In this example, we see an application of spectral clustering on image segmentation. We create four circles with centres (37, 28), (45, 52), (63, 56), and (26, 68) and radii 15, 12, 13, 14. We then form an image with all the four circles plotted together along with some background noise from $\mathscr{U}(1, 1.3)$ distribution. The figure generated can be seen in Figure (2).

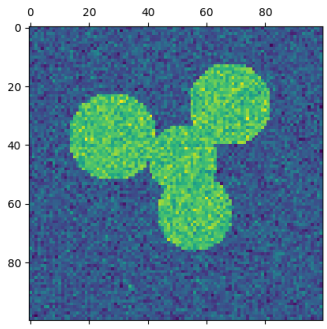


Figure 2: Four overlapping circles with background noise.

Using spectral clustering with the natural cluster number 4 and Gaussian similarity function, we get the figure in Figure (3).

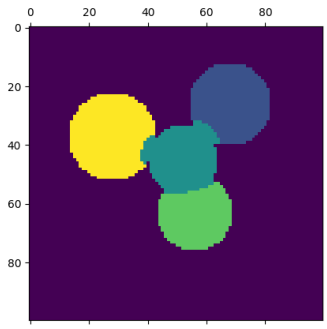


Figure 3: Spectral clustering segments the four circles into four clusters successfully.

We see that spectral clustering is successfully in segmenting the four overlapping circles even from the background noise present.

9.2 Real Data

9.2.1 Iris dataset

We perform Spectral clustering on Fisher's Iris dataset. Although based on Figure (4), we are getting an optimal value of the number of clusters k based on eigengap heuristic to be 2, we take the optimal number of clusters to be three and perform

normalized spectral clustering. We think there is something erroneous in our code, which we are unable to debug at the moment of writing this report. After fitting, we measure the accuracy of the assigned labels by

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of predictions}}.$$

We obtained the value of the Accuracy metric as 0.9 on the unscaled dataset. So, we can conclude that Spectral Clustering works quite well on this dataset, even if it is unscaled. We note that on fitting two components, we got an accuracy score of 0.67.

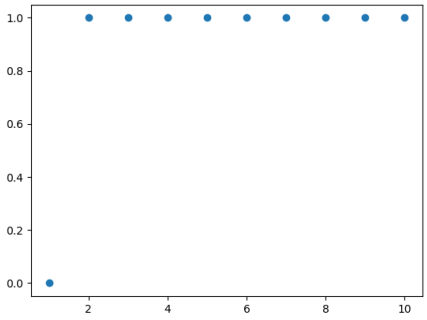


Figure 4: Plot of eigenvalues of symmetric graph Laplacian associated with the Iris dataset in an increasing order.

10 Supplementary Material

The interested reader is directed to https://github.com/ArkaB-DS/SpectralClustering which contains all the figures present here in the directory `images` and the corresponding codes to generate them in the `R` directory.

11 Acknowledgements

I take this opportunity to heartily thank my supervisor Prof. Amit Mitra for his valuable feedback and constant guidance on this project.

References

Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769.

Hagen, L. and Kahng, A. B. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Meilă, M. and Shi, J. (2001). A random walks view of spectral segmentation. In *International Workshop on Artificial Intelligence and Statistics*, pages 203–208. PMLR.

Mohar, B. (1997). Some applications of laplace eigenvalues of graphs. In *Graph symmetry*, pages 225–275. Springer.

Mohar, B., Alavi, Y., Chartrand, G., and Oellermann, O. (1991). The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12.

Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

Stoer, M. and Wagner, F. (1997). A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

Wagner, D. and Wagner, F. (1993). Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. Springer.