[WORK IN PROGRESS]
Research Report - Spectral Clustering

Sourabh Antani

Chapter 1

Introduction

Now a days, Clustering has become one of the most fundamental tasks in any project involving exploratory data analysis. Clustering helps the researchers understand the diversity of the data and number of different populations that exist. This knowledge, in turn, aids in subsequent tasks like choice modelling or classification techniques or parameters for tuning them.

In this report, I will list some of the commonly used clustering techniques before diving deeper into Spectral Clustring. Thereafter we will look at some of the common challanges with spectral clustering and some of the existing advancements that have been proposed to alleviate them. Finally I will list some of the possible avenues of research that I wish to research.

Clustering Algorithms

In practice, apart from Spectral Clustering, various types of clustering algorighms are used, ranging from the ones based on spatial distribution to matrix factorization to eigen-value based methods.

Some common examples of algorighms that rely on euclidean distance and spacial density of points are k-means (term first used by J. Macqueen [1], standard algorithm published by Lloyd [2]), which iteratively computes the controids of the clusters using euclidean distances, BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)[3], which extracts centroids from building Hierarchies, mean-shift[4] and DBSCAN [5], which locates the centroids by iteratively following the higher density areas. while these are generally simpler to understand, the 'shape' of the spatial distribution greately affects the effectivenes of these algorithms.

Non-Negative Matrix Factorization [6] [7] (later [8] studied the equivalence of NNMF with Spectral Clustering) and PCA Based clustering [9] are examples of techniques based on matrix factorization. These techniques seek to factorize the matrix and use one of the factors as approximate basis for the matrix, thus effectively reducing the dimentionality of the problem. Another advantage of these methods is that it simple to guage how 'good' the approximation is by looking at the difference between the product of approximated factors and the original matrix. Thus finding out the number of 'extracted' features or

reduced dimention that captures the desired variability in the data.

Chapter 2

Spectral Clustering

An excellent tutorial to Spectral Clustering can be found in [10]. Before we look at the various algorithms, let us go through some basic terms involved.

2.1 Some Terminology

Given an undirected graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ being the set of vertices and E being the set of weighted edges such that each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$.

The Weighted *adjacency matrix* of the graph is the matrix $W = (w_{ij})_{i,j=1,\ldots,n}$. If $w_{ij} = 0$, vertices v_i and v_j are not connected. $w_{ij} = w_{ji}$ since G is undirected.

The degree of a vertex $v_i \in V$ is defined as $d_i = \sum_{j=1}^{n} w_{ij}$. *degree matrix* D is defined as the diagonal matrix with the degrees $d_1, \ldots, d_n$ on the diagonal.

$\bar{A}$ denotes the Complement $V \backslash A$ of a given subset $A \subset V$. Indicator vector $\mathbf{1}_A$ is a vector with entries $f_i = 1$ if $v_i = A$, $f_i = 0$ otherwise.

Define $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$:], for not necessarily disjoint sets $A, B \subset V$. The 'size' of $A \subset V$ can be measured in two ways, $|A|$:= number of vertices in A or $vol(A) := \sum_{i \in A} d_i$

A subset $A \subset V$ of a graph is connected if any two vertices in A can be joined by a path such that all intermediate points also line in A. A is called Connected Component if it is a connected subset such that A and $\bar{A}$ are disjoint. FInally, Partitions of a graph are defined as non-empty sets $A_1, \ldots, A_k$ form partition of graph inf $A_i \cap A_j = \emptyset$ and $A_1 \cup \cdots \cup A_k = V$

2.2 Graph Laplacians and Graph cuts

2.2.1 Graph Laplacians

The following types of graph Laplacians have been defined in the literature:

$$\textbf{Unnormalized Graph Laplacian: } L = D - W$$
$$\textbf{Normalized Graph Laplacian (Symmetric): } L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$
$$\textbf{Normalized Graph Laplacian (Raondom walk): } L = D^{-1} L = I - D^{-1} W$$

All three graph Laplacians are positive-semidifinte and have non-negative real valued eigen values. Also, 0 is an eigenvalue with multiplicity equal to number of connected components of the graph. Thus for fully connected graph one of the eigenvalues is 0. L and L_{sym} are symmetric. The eigenvectors of L_{rw} are the eigen vectors of L while $D^{1/2}u$ is eigenvector of L_{sym} if u is eigenvector of L. For proofs of the above properties, refer to the appendix. As far as naming goes, L_{sym} follows from its structure. While L_{rw} is named because its structure is derived from the point of view of random walk on the graph. Taking $p_j = w_{ij}/d_i$ the probability of an edge between v_i and v_j being selected for random walk, the

transition matrix for the random walk is defined by $P = D^{-1}W$. Thus $L_{rw} = I - P$. This is why the second form of nomalized graph Laplacian is denoted by L_{rw}. For information about unnormalized graph Laplacian, refer [11] and [12] while the standard reference for normalized graph Laplacian is [13]. A formal equivalence between Ncut and transition probability of the random walk has been outlined in [14]

2.2.2 Graph Cuts

The intuition of clustering is to divide the graph in to groups of vertices such that he edges between the vertices in the same group have high weight and the edges between vertices from different groups have zero or very low weight. Thus effective clustering is to solve the minicut problem which can be defined as, for a given number k subsets, choosing the a partition $A_1, \ldots, A_k$ which minimizes

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

In order to make sure that minicut does not simply spearate an individual vertex, two approaches have been suggested to make sure that the clusters are 'reasonably large'.

First is RatioCut [15] where the solution is to solve minicut problem while dividing the graph in components with roughly the same number of vertices. The second approach is Ncut [16] which tries to solve the minicut problem by keeping the volume of each component, i.e. sum of edge weights, roughly the same.

Thus, the objective functions that we seek to minimize are

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

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

While the introduction of these balancing condition makes the mincut problem NP hard, relaxing these conditions slightly leads Ncut and RatioCut to normalized and unnormalized spectral clustering respectively [10]

2.2.3 Perturbation theory point of view

Perturbation theory states that the eigenvalues and eigenvectors of a matrix A and a perturbed matrix $A + H$ is bounded by a constant times the norm of H. To apply this to spectral clustering, consider the ideal case where the graph is composed by k connected components. In this case, the first k eigenvalues will be 0 and corresponding eigenvectors will be the indicator vector of the cluster, where exactly one entry (corresponding to the cluster) will be 1 and rest will be 0. In this case, k-means will trivially converge to the indicator vectors and clustering will be ideal. With a small perturbation, the eigenvectors will also be slightly perturbed but k-means should still converge to the correct centroids. However, if the noise or perturbation is large, or the non=zero eigenvalues are very close to 0 or eigengap is very small, thus producing eigenvectors whose original values are very close to 0, the perturbation may be large enough for k-means to predictably converge to the correct centroids. The problem mentioned in the last section is not common for L or L_{rw} but for L_{sym} the eigenvectors are already left multiplied by $D^{1/2}$ and can cause trouble. Hence the third algorithm below [17] needs an extra normalization step.

2.3 Algorithms

Three classic spectral clustering algorithms can be found in literature.

All three algorithms essentially follow the same steps, using the first k eigen vectors of the graph Laplacian, create a matrix and then create k clusters from the rows of that matrix using k-Means algorithm. Finally, create the clusters of data points with the same indices as the indices of the matrix rows in the

k clusters formed by k-means. For a proof of how the relaxation of the above balancing conditions to arrive at an approximation of NCut and RatioCut leads to the Normalized and Unnormalized Spectral Clustering respectively, see [10]

2.3.1 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 2. 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 $i = 1, \ldots, n$, 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 $A_i = \{x_j | y_j \in C_i\}$.

2.3.2 Normalized Spectral Clustering - Shi & Malik (2000)[16]

This algorithm uses generalized eigenvectors of L, which are the eigenvectors of normalized random-walk laplacian L_{rw}

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 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of generalized eigenproblem $Lu = \lambda Du$.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, 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 $A_i = \{x_j | y_j \in C_i\}$.

2.3.3 Normalized Spectral Clustering - Ng, Jordan & Weiss (2002)[17]

This algorithm uses the eigenvectors of normalized symmetric laplacian L_{sym}. Note that if $D^{1/2}u$ is an eigenvector of L_{sym} if u is an eigenvector of L hence an additional normalization step is needed.

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 2. 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 norm to 1, $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$
- For $i = 1, \ldots, n$, 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 $A_i = \{x_j | y_j \in C_i\}$.

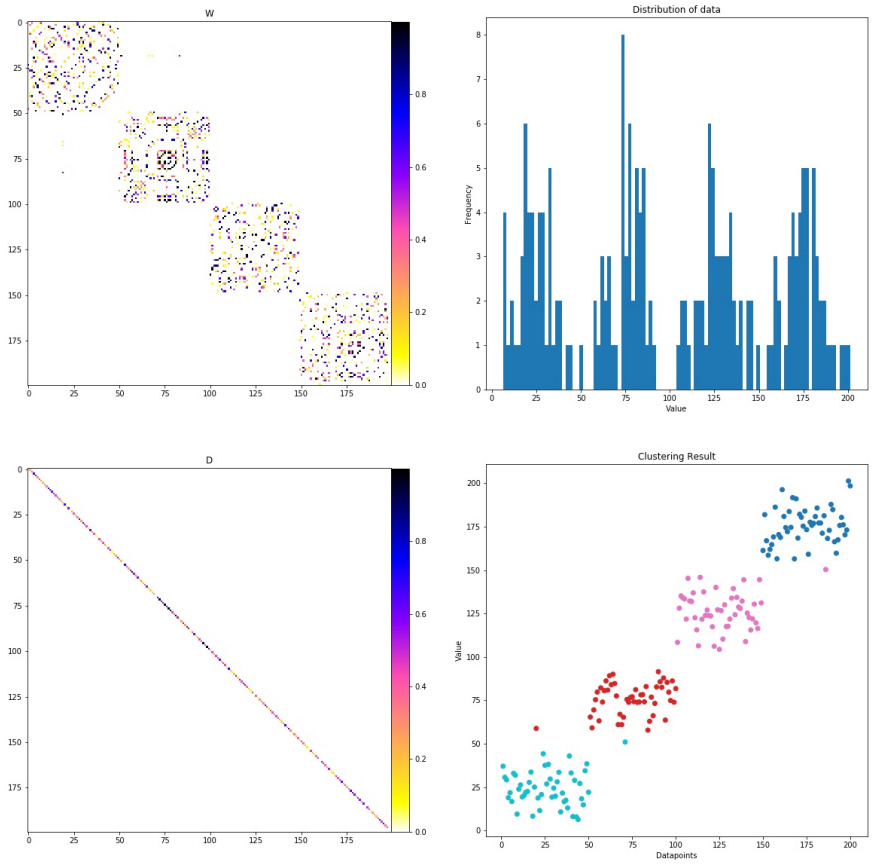

Figure 2.1: Clustering results for 200 points in $\mathbb{R}^1$

2.3.4 Numerical Experiments

Using my implementation of the above algorithms, here are some results of my experiments

1. Mixture of 4 Gaussians in $\mathbb{R}^1$
   This data set consists of a random sample of 200 points drawn according to a mixture of four Gaussians. The similarity function used was a gaussian kernel $e^{-|x_i - x_j|^2/(2\sigma^2)}$ with $\sigma = 1$. The distribution of points, pictorial representation of matrices W and D and the final clustering results are showin in Figure 2.1

2. Points in $\mathbb{R}^2$
   This data set consists of 1351 points in $\mathbb{R}^2$ distributed in 9 clusters with one outlier. The figure below shows the clustering in 5, 8 and 10 clusters. This example shows that even though in theory, both Ncut and RatioCut try to keep the clusters reasonably large, that if a single oulier point, if sufficiently separaed from the rest, is clustered independently. The reason for this is that rest of the clusters are very tightly organized. The similarity function used in this case was scaled euclidean distance. The clustering results in to 5, 8 and 10 clusters is shown in Figure 2.2

2.4 Practical considerations and challenges

While the algorithms are relatively simple to implement, there are some issues that arise in practice.

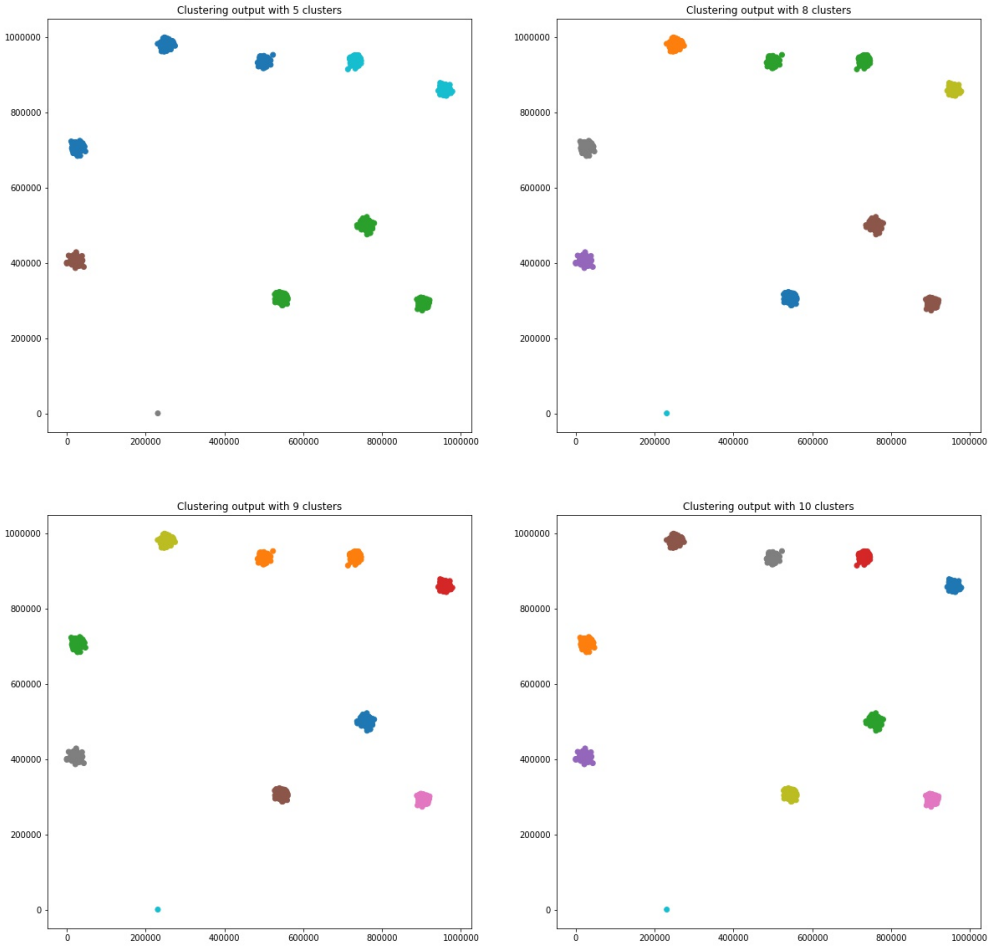

Figure 2.2: Clustering results for 200 points in $\mathbb{R}^1$

2.4.1 Parameters and Tuning Considerations

The first consideration is the choice of similarity function. The effectiveness of algorithem depends upon the similarity matrix W which is defined by the similarity function. Although Gaussian similarity function $exp(-||x_I - x_j||^2)/(2\sigma^2)$ is generally a reasonable choice in Euclidean space, the choice of similarity function would generally be guided by the domain of application.

Next would be the choice of similarity graph and its parameters. The type of graph (k-nearest neighbor, ϵ-neighborhood etc.) affects how the areas of different densities are treated. For example, k-nearest neighbor graph may cause some points in a sparse cluster to be connected to a dense cluster or may break a high density cluster into smaller clusters if k is not chosen appropriately. Also for ϵ neighborhood graph, the choice of ϵ dicates the sparsity of the matrix. While sparse matrix would be desirable since use of eigensolvers can greatly speedup the eigenvector calculation, it can also lead of loss of information and hence a balance must be achieved and choice of the parameters should be guided by the domain of application.

Choice of Laplacian is the next concern. Recall that the goal of clustering is two fold, to minimize inter-cluster similarity or $cut(A, \bar{A})$, and maximize intra-cluster similarity or $W(A, A)$. Both Ncut and RatioCut will achieve the first objective since they both have $cut(A, \bar{A})$ in the numerator. However, note that $W(A, A) = W(A, V) - W(A, \bar{A}) = vol(A) - cut(A, \bar{A})$. This means that intera-cluster similarity is maximized when $cut(A, \bar{A})$ is minimized and $vol(A)$ is maximized. This is where Ncut outperforms RatioCut since it has $vol(A)$ in the denominator. Hence L_{rw} is generally a good choice.

2.4.2 Challanges

The following are the challenges that frequently arise in practice.

Computing eigenvectors can be an expensive task, particularly if the matrix is large and dense. Fortunately a properly chosen graph and parameter(s) (e.g. k for k-means or ϵ for ϵ neighborhood), should leads to sparse matrix, which intern facilitates the choice of sparce eigensolvers and hence speed up the calculation. Additionally, the convergence speed depend upon spectral gap $\gamma_k = |\lambda_k - \lambda_{k+1}|$ and convergence depends upon the spectral radius of the matrix. Hence scaling and shifting or even methods like MAPS described in [18] can be applied.

Next, challenge would be to choose number of clusters. While the literature that describes the algorithms and their designs, usually takes the number of clusters, k, as input to the algorithms, frequently in practice, one needs to find the number of clusters that exist in a dataset. Since this is a general problem for all clustering algorithms, a variety of methods exist. Some methods use the ratio of intra-cluster to inter-cluster similarity or distance as a measure of how good the clustering is, while other methods use statistical calculation based on log-likelihood of data. Additinally, for spectral clustering, eigengap is an important heuristic. A good starting point is to choose k such that $\lambda_1, \ldots, \lambda_k$ are small but λ_{k+1} is relatively large. However, the effetiveness of this technique depends on how well the clusters are differentiated.

Finally, the k-means step itself can be expensive and in some cases, may not lead to ideal results based on the inital guess and/or the distribution of datapoints. Several attempts have been made to use other techniques of clustering. In [19], the authors show that k-means is efficiant and minimizes the quantization error, thus highlight the reasons why k-means has become the choice for clustering applications.

2.5 Compressive Spectral Clustering

In Compressive Spectral Clustering [20], the authors propose an algorithm to sample $\mathcal{O}(\log(k))$ randomly fitered signals on the graph to serve as feature vectors instead of eigenvectors and by clustering random subset of $\mathcal{O}(k \log(k))$ nodes using random feature vectors and infering the cluster label of all N nodes. This algorithm speeds up the clustering process by reducing the dimentionality and hence, the size of the problem.

As discussed above, for a graph with k connected components, 0 is eigenvalue with multiplicity k and the corresponding eigenvectors would be the indicator vectors of the clusters, which would be orthogonal. Thus, the indicator vectors belong to $spanU_k$. Perturbation theory says that a for the general graph

with slight perturbation to ideal case, the first k eigenvectors still belong to $spanU_k$. Thus it should be possible to sample a smaller number of features and still have k-means identify the k clusters correctly.

The altgorithm for compressive spectral clustering is, however quite involved and is outlined briefly below.

- First, to get the first k eigenvectors,

  Estimate the k^{th} eigenvalue using eigen counting techniques detailed in [21]

  Build polynomial estimation of the low-pass filter $H := h(L) = Uh(\Lambda)U^T$ where $h(\Lambda)$ is the diagonal matrix with 1 on primary diagonal for the first k rows/columns and 0 otherwise. U here is the matrix is the eigenvectors of L. However, L is not diagnolized, instead the lowpass filter Hx is approximated as $\sum_{l=0}^{p} \alpha_l L^l x$ by successive matrix-vector products with L.
- Generate d random Gaussian signals with mean 0 and variance $1/d$, $R = (r_1|r_2|\dots|r_d) \in \mathbb{R}^{N\times d}$
- Filter R with H and define for each node i, its feature vector $\bar{f}_i \in \mathbb{R}^d = \left[(HR)^T\delta_i\right]/||(HR)^T\delta_i||$
- Generate random sampling matrix $M \in \mathbb{R}^{n\times N}$ to keep $n = 2k\log k$ features.
- Run k-means on the reduced dataset with euclidean vector distance to obtain k reduced indicator vectors, one for each cluster.
- Interpolate each reduced indicator vector to find the cluster assignment for each node on the graph.

Chapter 3

Possible optimizations

In practice quite frequently the number of cluster is not known. Hence a clustering method is repeatedly applied with increasing number of clusters until desired accuracy is reached. With spectral clustering, since the number of clusters depends on the number of eigenvectors used, the information from a prior step may be used in ths subsequent step to avoid redundant work. For example, since the first k eigenvectors and hence, eigenvalues are, known, the next eigenvalue may be found by using shift inverse power iteration.

Another possibility is to use Krylov subspace based methods. Although I do not have a clear path, Krylov space methods are attractive possibility for two reasons, one is that adding dimentions, i.e. subsequent basis vectors is simply matrix-vector product and second reason is that L and L_{sym} are symmetric and thus will lead to Lanczos method and tridiagonal instead of Hessenberg matrix. It may be possible to

achieve clustering using Krylov spbspace basis vectors instead of eigenvectors.

Appendix A

Propositions related to Graph Laplacians

Proposition 1. *(Properties of L) The matrix L satisfies:*

1. *for every vector* $f \in \mathbb{R}^n$ *we have*

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

2. *L is symmetric and positive semi-definite*
3. *The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant vector* $\mathbf{1}$
4. *L has n non-negative, real-values eigenvalues* $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

Proof:

1. By definition of $d_j = \sum_{i=1}^{n} w_{ij}$

$$\begin{aligned} f'Lf = f'Df - f'Wf &= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2}\Big(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{i=j}^{n} d_j f_j^2 \Big) \\ &= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \end{aligned}$$

2. Since W and D are symmetric $L = D - W$ is still symmetric. By result of part (a), since $w_{ij} \geq 0$ and $(f_i - f_j)^2 \geq 0$, $f'Lf \geq 0$, hence L is symmetric and positive semi-definite
3. See proposition 2
4. Since L is symmetric, all its eigenvalues are real. Also by part (c), the smallest eigenvalue is 0,

Note that the unnormalized graph Laplacian does not depend on the diagonal elements of W. $d_{ii} = \sum_{j=1}^{n} w_{ij}$, hence the diagonal elements are cancelled by d_{ii} and the self-edges do not change the Laplacian.

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

Proof: Starting with $k = 1$, i.e. graph is fully connected. Assume that f is the eigenvector with eigenvalue 0. Then,

$$0 = f'Lf = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \implies f_i - f_j = 0 \quad [\because w_{ij} > 0]$$

Since we assumed a fully connected graph, f has to be constant for the above argument to be true for any vertex in the connected component. Hence the vector $\mathbf{1}$, which is the indicator vector of the component is an eigenvector corresponding to eigenvalue 0.

Now, for $k > 1$, we can easily rearrange the vertices in the matrix to put all the vertices of a connected component together. This would cause the matrix L to be a block diagonal matrix. For each of these blocks, or components, the respective indicator vector is an eigenvector with eigenvalue 0. Thus the Matrix L has eigenvalue 0 with multiplicity equal to the number of components and the corresponding eigenvectors would be the indicator vectors of those components.

Proposition 3. *(Properties of L_{sym} and L_{rw})*

1. *For every $f \in \mathbb{R}^n$*
$$f'L_{sym}f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$
2. *λ is an eigenvalue of L_{rw} with eigenvector u iff λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}u$.*
3. *λ is an eigenvalue of L_{rw} with eigenvector u iff λ and u solve the generalized eigenproblem $Lu = \lambda Du$*
4. *0 is an eigenvalue of L_{rw} with the constant one vector $\mathbf{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbf{1}$*
5. *L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$*

Proof:

1. This can be proved similar to part 1 of proposition 1.
2. Assume λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}u$.
$$\begin{aligned} & L_{sym}D^{1/2}u = \lambda D^{1/2}u \\ \implies & D^{-1/2}LD^{-1/2}D^{1/2}u = \lambda D^{1/2}u \\ \implies & D^{-1}Lu = \lambda u \quad [\text{left multiplying with } D^{-1/2}] \\ \implies & L_{rw}u = \lambda u \end{aligned}$$
In otherwords, u is eigenvector of L_{rw}. To prove the other direction, apply the above steps in reverse order.
3. This can be proven in similar manner to above by left multiplying the generalized eigenproblem $Lu = \lambda Du$ with D^{-1}
4. $L_{rw}\mathbf{1} = (I - D^{-1}W)\mathbf{1} = \mathbf{1} - \mathbf{1} = 0$.
The above is true because $d_i = \sum_{j=1}^{n} wij \implies \left(D^{-1}W)\mathbf{1}\right)_i = \sum_{j=1}^{n} \frac{w_{ij}}{d_i}$, since $\mathbf{1}$ is the indicator vector. Then using part (b) leads to the second statement of this property.
5. The first part (for L_{sum}) can be proved from part (a), similar to Proposition 1. Part 2 essentially states that L_{sym} and L_{rw} have the same eigenvalues, hence second statement is also proven.

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

Proof: The proof of this is analogous to that of proposition 2, but using proposition 3.

Bibliography

[1] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[2] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[3] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA, 1996. Association for Computing Machinery.

[4] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 1995.

[5] Martin Ester, Hans-Peter Kriegel, JÃ¶rg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[6] William H. Lawton and Edward A. Sylvestre. Self modeling curve resolution. *Technometrics*, 13(3):617–633, 1971.

[7] Pentti Paatero, Unto Tapper, Pasi Aalto, and Markku Kulmala. Matrix factorization methods for analysing diffusion battery data. *Journal of Aerosol Science*, 22:S273–S276, 1991. Proceedings of the 1991 European Aerosol Conference.

[8] Chris Ding, Xiaofeng He, and Horst D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *in SIAM International Conference on Data Mining*, 2005.

[9] Nian Zhang, Keenan Leatham, Jiang Xiong, and Jing Zhong. Pca-k-means based clustering algorithm for high dimensional and overlapping spectra signals. pages 349–354, 11 2018.

[10] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[11] B Mohar, Y Alavi, G Chartrand, Ortrud Oellermann, and Allen Schwenk. The laplacian spectrum of graphs. *Graph Theory, Combinatorics and Applications*, 2:5364, 01 1991.

[12] Bojan Mohar. *Some applications of Laplace eigenvalues of graphs*, pages 225–275. Springer Netherlands, Dordrecht, 1997.

[13] F. R. K. Chung. *Spectral Graph Theory.* Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[14] Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. 2001.

[15] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9), 1992.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[17] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Proc. NIPS*, 2001.

[18] Songtao Lu and Zhengdao Wang. Accelerated algorithms for eigen-value decomposition with application to spectral clustering. *2015 49th Asilomar Conference on Signals, Systems and Computers*, pages 355–359, 2015.

[19] LÃ©on Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press, 1995.

[20] Nicolas Tremblay, Gilles Puy, Remi Gribonval, and Pierre Vandergheynst. Compressive spectral clustering, arXiv:1602.02018, 2016.

[21] Edoardo Di, Napoli Eric, and Polizzi Yousef Saad. Efficient estimation of eigenvalue counts in an interval, 2013.

[22] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[23] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Number 20 in Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1998.

[24] G. W. Stewart and J. G. Sun. *Matrix perturbation theory*. Academic Press, Boston, MA, 1990.

[25] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng. Graph spectral image processing. *Proceedings of the IEEE*, 106(5):907–930, 2018.