[WORK IN PROGRESS]
Report on Paper ”A tutorial on spectral clustering” by Ulrike von Luxburg

Sourabh Antani

Chapter 1

Introduction

This is a paper report on the paper ’A tutorial on spectral clustering’ by Ulrike von Luxburg [1]. All the sections in this report are taken or paraphrased from [1] and not original work. The python code in the github repository at https://github.com/sourabhantani-smu/research is based on the algorithms stated and described in the paper [1].

This tutorial is setup up as self-contained introduction to spectral clustering. It derives spectral clustering from scratch and present different points of view to why spectral clustering works. After describing the some of the basic terms in the in beginning of the paper, the authors describe the 3 classic algorithms for unnormalized and normalized spectral clustering. After describing the algorithms and the results on a simple test problem, the authors dive deeper to derive the three algorithms and provide the intuition behind each as applicable to graph with connected components. Later, the authors describe why the same technique would work even if the entire graph is connected. Finally the authors present some challenges that one may face while applying the algorithms, and how to choose the correct algorithm and parameters their of.

Chapter 2

Similarity graphs

Given a set of data points $x_1, \dots, x_n$ and some notion of similarity $s_{ij} \geq 0$ getween all pairs of data points x_i and x_j, the intuitive goal of clustering is to divide the data points into several groups such that the points in the same group are similar and the points in different groups are dissimilar to each other. A nice way of representing the data is in form of *similarity graph* $G = (V, E)$. Each vertex $v_i$in this graph represents a data point x_i. Two vertices are connected if the similarity s_{ij} between the corresponding data points x_i and x_j is positive or larger than a certain threshold and the edge is weighted by s_{ij}. The problem of clustering can now be reformulated using the similarity graph: we want to find the partition of the graph such that the edges between different groups have very low weights (which points to dissimilarity) and the edges within the group have high weights (hence similar).

2.1 Some Terminology:

$G = (V, E)$ undirected graph

$V = \{v_1, \ldots, v_n\}$ vertex set

E set of weighted edges such that each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$.

Weighted *adjacency matrix* of the graph is 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.

$\bar{A}$ Complement $V \backslash A$ of a given subset $A \subset C$.

Indicator vector $\mathbf{1}_A$ vector with entries $f_i = 1$ if $v_i = A$, $f_i = 0$ otherwise.

$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$, for non-necessarily disjoint sets.

$|A|$ = number fo vertices in A = size of set A

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

Connected subset $A \subset V$ of a graph is connected of any two vertices in A can be joined by a path such that all intermediate points also line in A.

Connected Component Connected subset such that A and $\bar{A}$ are disjoint

Partition 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 Different similarity graphs

ϵ-neighbourhood graph Connect points whose pairwise distances are less than ϵ. Weight data is not incorportated into the graph.

k-nearest neighbor graphs Connect the nearest k points. Usually directed since v_i being in the k *nearest neighbors* of v_j does not mean that v_j is in k *nearest neighbors* of $v_(i)$

Fully connected graph is a graph where all points with positive similarity with each other are connected. The edges here are weighted by s_{ij}. This type of construction is only useful when the similarity function models local neighborhoods since a graph should represent local neighborhood relationships.

Chapter 3

Graph Laplacians and their basic properties

Assumptions: G is an undirected, weighted graph with weight matrix W, where $w_{ij} = w_{ji} \geq 0$. Eigenvectors are not normalized and multiples of an eigenvector are considered to be equivalent. Eigenvalues are always ordered increasingly, respecting multiplicities. So ’the first k eigenvectors’ will refer to eigenvectors corresponding to the k smallest eigenvalues.

3.1 Unnormalized graph Laplacian: $L = D - W$

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 $d_j = \sum_{i=1}^{n} w_{ij}$

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

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.

3.2 Normalized Graph Laplacians

There are two normalized graph Laplacians found in literature:

1. Symmetric Laplacian: $L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$
   This can be derived simply by decomposing the diagonal matrix D into $D^{1/2}D^{1/2}$ and then left and right multiplying the equation $L = D - W = D^{1/2}D^{1/2} - W$ by $D^{-1/2}$
2. Random walk Laplacian: $L_{rw} = D^{-1}L = I - D^{-1}W$
   This can be derived by left multiplying $L = D - W$ by D^{-1}

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 $\mathbf{1_{A_i}}$ of these components. for L_{sym} the eigenspace of 0 is spanned by $D^{1/2}\mathbf{1_{A_i}}$*

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

Chapter 4

Spectral clustering algorithms

This chapter lists 3 classic algorithms, one for unnormalized and two for normalized spectral clustering. These have been implemented in python notebooks on the git repository and are excluded here.

Chapter 5

Graph cut point of view

The intuition of clustering is to separate points into groups so that the edges between groups have low weight. Thus the clustering can be viewed as mincut problem.

Recall the notation $W(A,B) = \sum_{i\in A, j\in B} w_{ij}$ and $\bar{A}$ for compement of A. Thus choosing the k partitions $A_1, \dots, A_k$ is to minimize

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

The factor 1/2 is for notational consistency, otherwise the edges in the cut will be counted twice, once from A to $\bar{A}$ and then from $\bar{A}$ to A. The problem is that in many cases this separates in to 'unbalanced' clusters where some clusters have only one (or very few) nodes while others are fairly large. One way to circumvent is to explicitly request that the sets $A_1, \dots, A_k$ are 'reasonably large'. Two common approaches are RatioCut (Hagen & Kahng, 1992) where the size of A_i is measured by the number of vertices $|A|$ and Normalized cut, or Ncut, (Shi & Malid 2000) where the size A_i is measured by sum of weights of its edges $vol(A)$. The definitions are

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

Note that in both cases, the objective functions take the minimum value if the 'measures' (number of vertices or sum of weights) for all the subsets(i.e. partitions/clusters) coincide (are reasonably close), thus aiding in generating 'balanced' clusters.

Unfortunately adding this 'balancing' condition makes the problem NP-hard. Therefore in practice this is achieved by relaxing the definitions and that is what spectral clustering achieves.

5.1 Approximating RatioCut for $k = 2$

The goal is to solve the optimization problem

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

Given $A \subset V$, define $f = (f_1, \dots, f_n)' \in \mathbb{R}^n$ with entries

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|}, & \text{if } v_i \in A, \\ \sqrt{|A|/|\bar{A}|}, & \text{if } v_i \in \bar{A} \end{cases}$$

Now, we can write the RatioCut using Unnormalized graph Laplacian

$$\begin{aligned} f'Lf &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \\ &= \frac{1}{2}\sum_{i\in A, j\in \bar{A}} w_{ij}\left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right) \\ &\quad + \frac{1}{2}\sum_{i\in \bar{A}, j\in A} w_{ij}\left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}}\right) \\ &= \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i)\left(\frac{|\bar{A}_i|}{|A_i|} + \frac{|A_i|}{|\bar{A}_i|} + 2\right) \\ &= cut(A, \bar{A})\left(\frac{|A| + |\bar{A}|}{|A|} + \frac{|A| + |\bar{A}|}{|\bar{A}|}\right) \qquad \left[\because 2 = \frac{|A|}{|A|} + \frac{|\bar{A}|}{|\bar{A}|}\right] \\ &= |V| \cdot RatioCut(A, \bar{A}) \qquad \left[\because |A| + |\bar{A}| = |V|, \frac{cut(A, \bar{A})}{|A|} + \frac{cut(A, \bar{A})}{|\bar{A}|} = RatioCut(A, \bar{A})\right] \end{aligned}$$

Also,

$$f \cdot \mathbf{1} = \sum_{i=1}^{n} f_i = \sum_{i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i \in \bar{A}} \sqrt{\frac{|A|}{|\bar{A}|}} = |A|\sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}|\sqrt{\frac{|A|}{|\bar{A}|}} = 0 \implies f \perp \mathbf{1}$$

$$||f||^2 = \sum_{i=1}^{n} f_i^2 = |A|\frac{|\bar{A}|}{|A|} + |\bar{A}|\frac{|A|}{|\bar{A}|} = |A| + |\bar{A}| = n$$

Thus the goal is to find such a vector f that minimizes $f'Lf$ under the conditions $f \perp \mathbf{1}$ and $||f|| = \sqrt{n}$. Here n is the total number of vertices in the graph, hence is a constant in this equation. This means that the vector f that minimizes $f'Lf$ will also minimize $\frac{f'Lf}{n} = \frac{f'Lf}{||f'f||}$.

Rayleigh-Ritz ([2]) theorem states that

> Let $A \in \mathbb{C}$ be a Hermitian Matrix, and 'Rayleigh quotient' be the function defined as
>
> $$R : \mathbb{C}^n \backslash \{\mathbf{0}\} \to \mathbb{R}, R(x) = \frac{x^H A x}{x^H x}, ||x|| \neq 0$$
>
> then the eigenvectors of A are the critical points of $R(x)$ and eigenvalues of A are the values of $R(x)$ at those critical points.

Consequently,

$$\lambda_{max} = \max_{||x|| \neq 0} \frac{x^H A x}{x^H x} \qquad \text{and} \qquad \lambda_{min} = \min_{||x|| \neq 0} \frac{x^H A x}{x^H x}$$

Thus, f would be the eigenvector corresponding to the smallest eigenvalue of L. However, we know that the smallest eigenvalue of L is 0 with eigenvector as $\mathbf{1}$. Hence, we can use the second eigenvector of L as the minimizer of the RatioCut function. This is just the solution to the 'relaxed' problem. Hence to find the partition of the graph, we need to transform this real-valued solution into discrete solution. The simplest way to do this is to use the sign of the f as the indicator function. i.e. split the graph in to A and $\bar{A}$ by A is set of the vertices where the corresponding element is non-negative and $\bar{A}$ is the remaining. i.e.

$$\begin{cases} v_i \in A, & f_i \geq 0 \\ v_i \in \bar{A} & f_i < 0 \end{cases}$$

In practice, what most spectral clustering algorithms do, instead is to treat the elements (coordinates) of f as points in $\mathbb{R}$ and use $k-$means to cluster them into two clusters (say C_1 and C_2) and then carry over the clustering to the graph by choosing points as

$$\begin{cases} v_i \in A, & f_i \in C_1 \\ v_i \in \bar{A} & f_i \in C_2 \end{cases}$$

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

5.2 Approximating RatioCut for arbitary K

Generalizing the above method, we partition the set of vertices V into k sets $A_1, \ldots, A_k$. We define k indicator vectors $h_j = (h_{1,j}, \ldots, h_{n,j})'$ by

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

Let $H \in \mathbb{R}^{n \times k}$ be the matrix of the k indicator vectors above. Note that the columns of H are orthonormal to eachother, i.e. $H'H = I$. Applying the calculations similar to the last section we find that

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

Thus,

$$RatioCut(A_1, \dots, A_k) = \sum_{i=1}^{k} h_i' L h_i = \sum_{i=1}^{k} (H'LH)_i = Trace(H'LH)$$

Hence the goal of clustering would be minimize Trace of $H'LH$ with H orthonormal as defined above. Again, a version of Rayleigh-Ritz theorem ([2]) states that this is achieved by taking the first k eigenvectors of L corresponding to the k smallest eigenvalues. Note that H is the same as matrix U in unnormalized spectral clustering mentioned in chapter 4. Again converting this to the discretion partition, we get the unnormalized spectral clustering algorithm mentioned in chapter 4.

5.3 Approximating Ncut

Similar to RatioCut, for $k = 2$ we define cluster indicator vector f by

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

Similar to RatioCut, we can check that $(Df)'\mathbf{1} = 0$ and $f'Df = vol(V)$. Thus $f'Lf = vol(V)Ncut(A, \bar{A})$. So the problem is now converted to minimizing $f'Lf$ subject to f as defined above such that $Df \perp \mathbf{1}$ and $f'Df = vol(V)$.

Again, relaxing the problem by allowing to take arbitrary values, we substitute $g = D^{1/2}f$. Now the minimization problem becomes

$$\min_{g \in \mathbb{R}^n} g' D^{-1/2} L D^{-1/2} g, \qquad \text{such that } g \perp \mathbf{1}, ||g||^2 = vol(V)$$

Since $D^{-1/2}LD^{-1/2} = L_{sym}$, the problem becomes, again to minimize $g'L_{sym}g$ with $||g|| = \sqrt{vol(V)}$. Once again, Rayleigh-Ritz theorem states that the minimizer is the eigenvector of L_{sym}. We know that the first eigenvector is $D^{1/2}\mathbf{1}$ and constant eigenvalue is $vol(V)$, the solution of this minimization problem is given by second eigenvector of L_{sym}. Re-substituting $f = D^{-1/2}g$ and using parts (b) and (c) of proposition 3, we can state that, since $g = D^{1/2}f$ is an eigenvector of L_{sym}, letting λ be the corresponding eigenvalue, f is eigenvector of L_{rw} and solves the generalized eigenvalue problem $Lu = \lambda Du$.

Extending this for $k > 2$, let

$$h_{i,j} = \begin{cases} 1/\sqrt{vol(A_j)} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases} \quad (i = 1, \dots, n; j = 1, \dots, k)$$

Now, set the matrix H as the matrix containing first k indicator vectors as columns. Observe that $H'H = I$, $h_i'Dh_i = 1$ and $h_i'Lh_i = cut(A_i, \bar{A}_i)/vol(A_i)$. Hence the minimization problem now becomes the trace minimization problem for $Trace(H'LH)$ subject to $H'DH = I$. Relaxing the discreteness condition and substituting $T = D^{1/2}H$, we get $minT \in \mathbb{R}^{n \times k} T' L_{sym} T$, subject to $T'T = I$. Once again, combining the Rayleigh-Ritz theorem as applied to RatioCut, and Proposition 3 parts (b) and (c) we can see that the solution is to construct H with the first k eigenvectors of L_{rw}, or the first k generalized eigenvectors of $Lu = \lambda Du$. This yields the normalized spectral clustering algorithm according to Shi and Malik (2000) which is also mentioned in chapter 4, and implemented in a jupyter notebook on github ripository.

5.4 Comments on relaxation approach

The author notes that the relaxation approach leads to an approximation of the real solution. There are no guarantees about the quality of the cut generated by these algorithms. For example, in the cockroach graph (which looks like a horizontal ladder with few rungs removed), the ideal cut will be vertical where the rungs have been removed. (need to get graphic in here) but the unnormalized spectral cut always generates horizontal cut. Here the ideal RatioCut = 1 but unnormalized clustering generates $2/k$, thus order of n factors worse. it is known that an algorithm that generates a constant RatioCut does not exist.

The relaxation techniques used here are not unique and many such techniques have been devised, mainly because they usually generated a standard Linear algebra problem to solve.

Chapter 6

Random walks point of view

A random walk on a graph is a stochastic process which randomly jumps from vertex to vertex. This section interprets spectral clustering as attempt to find the partition of the graph such that a random walk stays within a partition as long as possible. This makes more sense if we define the probability of jumping to a node to depend upon the edge weight between the two nodes. This would mean that the random walk would stay in the partition if the edges of the least probability are cut. Formally, the transition probability of jumping from v_i to v_j is given by $p_{ij} = w_{ij}/d_i$, i.e. the proportion of total degree of node v_i that is contributed by edge w_{ij}. Thus transition matrix $P = (p_{ij})$ is $P - D^{-1}W$. If the graph is connected and non-bipartite, then the stationary distribution (probability distribution that does not change from one step to next) of the random walk is always unique and can be represented as $\pi = [\pi_1, \ldots, \pi_n]'$ where $\pi_i = d_i/vol(V)$.

Recall that graph laplacian of random walk $L_{rw} = I - D^{-1}W = I - P$. This means that if (λ, u) is an eigenpair of L_{rw} then $L_{rw}u = \lambda u \implies (I-P)u = \lambda u \implies Pu = (1-\lambda)u$. Thus $(1-\lambda)$ is an eigenvalue of P and u is the corresponding eigenvector. This means that the eigenvectors corresponding to largest eigenvalues of P and consecutively, the smallest eigenvalues of L_{rw} can be used to describe the cluster properties of the graph.

6.1 Random walks and Ncut

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

Proof:

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

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

The final result them follows from definition of Ncut.

Thus, we can see that minimizing Ncut will reduce the probability of random walk transitioning from one subset to its complement, hence traversing from one cluster to next.

6.2 The commute distance

Commute distance between two vertices is defined as the time taken to travel from once vertex to the other and back. Hence reducing the commute distance is not a problem to find shortest path, but to find several short ways. This happens if the points involved in the path will be located in the same high-density region. Thus commute distance is very well-suited to clustering. Here the 'short' distance between two points is equivalent to points being close, hence similar and hence the edge weight being high.

Commute distance can be computed with help of generalize inverse of the graph Laplacian L. Recall that L is diagonalizable, hence $L = U\Lambda U'$, where U is matrix of eigenvectors as columns and Λ is the diagonal vector of eigenvalues. Since 0 is the smallest eigenvalue of L, it is not invertable. Hence we define $L^\dagger = U\Lambda^\dagger U'$ where $\Lambda^\dagger$ is diagonal matrix with entries $1/\lambda_i$ if $\lambda_i \neq 0$ and 0 if $\lambda_i = 0$. Thus entries of $L^\dagger$ are $l^\dagger_{ij} = \sum_{k=2}^{n} \frac{1}{\lambda_k} u_{ik} u_{jk}$ and it is symmetric positive semi-definite.

Proposition 6. *(Commute distance) Let $G = (V, E)$ a connected, undirected graph. Denote by c_{ij}, the commute distance between vertex v_i and vertex v_j, and by $L^\dagger = (l^\dagger_{ij=1...n})$ the generalized inverse of :, then we have $c_{ij} = vol(V)(l^\dagger_{ii} - 2l^\dagger_{ij} + l^\dagger_{j}) = vol(V)(e_i - e_j)'L^\dagger(e_i - e_j)$*

This proof shows that $\sqrt{c_{ij}}$ can be considered as eucledian difference. Spectral clustering here can be achieved by using proposition 6, to calculate $c_{ij} = vol(V)||z_i - z_j||^2$ where z_i is the point in $\mathbb{R}^n$ corresponding to i-th row of $U(\Lambda^\dagger)^{1/2}$.

This approach differs considerably in following ways:

1. In spectral clustering we map vertices of the graphs on rows of the matrix U while in commute time we map vertices on rows on $(\Lambda^\dagger)^{1/2}U$, thus scaling them by inverse eigenvalues of L.
2. Spectral clustering considers first k columns of matrix while commute time considers all columns.
3. If a graph has k disconnected components, first k eigenvalues will be 0 and first k columns of U are the indicator vectors of the clusters. However the first k columns of $(\Lambda^\dagger)^{1/2}U$ are zeros because of the zero eigenvalues. Hence the portion of the matrix that contains the useful information in both methods are completely different.

Note, however that if the graph is fully connected, then there is one eigenvector with eigenvalue 0 and in this case, this vector is ignored in both cases. Also the eigenvectors related to small eigenvalues are amplified since they are multiplied by $1/\lambda_i$. Hence both techniques behave similarly.

The author states that the relation between spectral clustering and commute time seems to be loose but helpful to understand the intuition. However it may be possible, in some cases, to tighten the relationship in some cases like the similarity function being strictly positive definite.

Chapter 7

Perturbation theory point of view

Perturbation theory states that the distance between a matrix A and $A + H$, where H is a small perturbation is bounded by a constant multiple of the norm of H. This constant depends on the eigenvalue and how far is the eigenvalue from rest of the spectrum. The basic idea here is that in the case where graph has k connected clusters, the first k eigenvectors of L or L_{rw} are indicator vectors of clusters and k-means algorithm would be able to trivially group the clusters. In a nearly-ideal case where the inter-cluster similarity is not exactly zero, the perturbation theory tells us that the eigenvectors will still be very close to the ideal indicator vectors. Hence, if the perturbation is not too large, i.e. the clusters are not very similar to each other, the k-means algorithm will still be able to separate the points into clusters.

7.1 The formal perturbation argument

The formal basis of this approach is the Davis-Kahan theorem stated below. For a bit of background, in perturbation theory, the distances between subspaces is usually measured in 'canonical angles' a.k.a. 'principle angles'. Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two p-dimensional subspaces of $\mathbb{R}^d$, V_1 and V_2 be two matrices such that columns form the orthonormal basis of $\mathcal{V}_1$ and $\mathcal{V}_2$ respectively. Then $\cos\Theta_i$ of the principle angles Θ_i are the singular values of $V_1'V_2$. Note that for one-dimensional case, canonical angles are the angles between the two systems and that the canonical angles can be defined even if the dimensions of two matrices are not the same. In this case the number of angles defined would be equal to the smaller number of dimensions of the two matrices.

Theorem 1. *(Davis-Kahan) Let A, $H \in \mathbb{R}^{n\times n}$ be symmetric matrices, and let $||\cdot||$ be the Frobenius norm or the two norm of matrices, respectively. Consider $\tilde{A} = A + H$ as a perturbed version of S. Let $S_1 \subset \mathbb{R}$ be an interval. Denote by $\sigma_{S_1}(A)$ the set of eigenvalues of A which are contained in S_1 and by V_1 the eigenspace corresponding to all those eigenvalues. Denote by $\sigma_{S_1}(\tilde{A})$ and $\tilde{V}_1$ the analogous quantities for $\tilde{A}$. Define the distance between S_1 and the spectrum of A outside of S_1 as*

$$\delta = min(|\lambda - s|); \lambda \text{ eigenvalue of } A, \lambda \notin S_1, s \in S_1$$

Then the distance $d(V_1, \tilde{V_1}) = ||\sin\Theta(V_1, \tilde{V_1})||$ between two subspaces V_1 and $\tilde{V}_1$ is bounded by

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

The author points to Section V.3 of Stewart and Sun (1990)[3]. To understand what the theorem means to spectral clustering, Let A be the graph Laplacian L of the ideal case where graph has k connected components and $\tilde{A}$ be the perturbed case where the k compnents are no longer completely disconnected but are connected by a few edges with low weight. Let the graph Laplacian of the 'lightly connected' case be $\tilde{L}$. Let $\lambda_1, \ldots, \lambda_n$ and $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ be the eigenvalues of L and $\tilde{L}$ respectively. We have to choose S_1 such that $\lambda_1, \ldots, \lambda_k$ and $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k$ are contained in S_1. This is easier when $H = L - \tilde{L}$ is small and $|\lambda_k - \lambda_{k+1}|$ is large. If we can find such a set (interval) S_1, the Davis-Kahan theorem states that the distance between the eigenspaces of the first k eigenvectors of L and $\tilde{L}$ would be bounded by $||H||/\delta$. Hence even for $\tilde{L}$, the spectral clustering using first k eigenvectors will find appropriate k clusters. How 'appropriate' the clustering is depends upon how large $|\lambda_k - \lambda_{k+1}|$ is.

It is worth noting that if $||H||$, which corresponds to the noise or inter-cluster similarity in the graph, is large we may not be able to find S_1 that has k eigenvalues both of L and $\tilde{V}$. In this case, the algorithm may end up using few more or less than k eigenvectors, $\tilde{k}$ then the ideal case, k and result in lower quality of clustering. The same may happen if the eigenvalues λ_k and λ_{k+1} are close to each other.

7.2 Comments about perturbation approach

Given that perturbation theory explains why the algorithms designed for ideal case of k connected components would also work for graphs where clusters are not completely disconnected, we have to be cautious. For example, many authors use the first k eigenvectors of similary matrix S or weight matrix W to discover clusters. Given the fact that any block diagonal symmetric matrix has the nice property that there is a basis of eigenvectors that will be zero outside a given block and real in the block, being block diagonal for ideal case is only a necessary condition for use of these eigenvectors for clustering. However, for successful use of these eigenvectors, at least two more properties should be satisfied:

1. We need to make sure that order of the eigenvalues and eigenvectors are meaningful. For Laplacian this is always true. Since we know that any connected component will have exactly one eigenvector whose eigenvalue 0, we know that first k eigenvectors of the Laplacian correspond to the k connected components of the graph, with one eigenvector per component. However this is not always true

for S or W. The two largest eigenvalues of W may come from the same block. In this case taking the k eigenvector would mean some blocks are represented more than once and others are not represented at all.

2. Second property that must be satisfied is that in the ideal case, the entries of the eigenvectors on the components should be 'safely bounded away' from 0. In the ideal case a non-zero entry will indicate that the point is a member of the respective cluster. However due to noise, if two entries of the eigenvectors of the perturbed matrix may be small values say $\tilde{u}_{1,i} = \epsilon_1$ and $\tilde{u}_{1,j} = \epsilon_2$, in this case it is unclear if the point should be considered to be a member of cluster i or j. This may still be a small problem for L and L_{rw}, but for L_{sym} the eigenvectors are not $\mathbf{1}_{A_i}$ but $D^{1/2}\mathbf{1}_{A_i}$. The algorithm is then very sensitive to the 'noise', especially if some of the nodes have very low degree. Hence sometimes row-normalization' will be applied to the matrix U in the algorithm of Ng et al. (2002). however, for the two small component case (with ϵ_1 and ϵ_2) both the compnents would be multiplied by $1/\sqrt{\epsilon_1^2 + \epsilon_2}$ and be very large, thus masking the real indicator compnents. On the other hand, some argue that a node with very low degree would be considered an outlier and should not be taken into account while judging the quality of clustering.

The discussion above shows that unnormalized spectral clustering (with L) and normalized spectral clustering with Random walk Laplacian (Lrw) are well justified but normalized spectral clustering with L_{sym} should be used with caution, especially if the graph contains vertices with very low degree.

Chapter 8

Practical details

This section discusses some of the issues that come up while implementing the spectral clustering algorithms.

8.1 Similarity Graph

8.1.1 Similarity Function

The similarity function itself has to be 'meaningful' because it induces the local neighborhoods formed in the graph. What it means for two points to be 'similar' is dependent on the domain and the application of the data. For example, similarity function between people would be very different from that between documents. Even within documents, the similarity would different based on the type of categorization needed (by author, title, geographical region of origin, historical era of origin etc). For data in Eucledian space, Gaussian similarity function $exp(-||x_i - x_j||^2)/(2\sigma^2)$ is a reasonable candidate but still the choice of parameter σ may prove crucial. It should be noted that the long-range behavior (the value of similarity returned for less similar or dis-similar data points) is less important because will eventually not connect these points in similarity graph anyways.

8.1.2 Type of similarity graph

Next choice is to choose the graph type, k-nearest neighbor, ϵ-neighborhood etc.

The paper shows using a data set of two 'moons' and a Gaussian how ϵ-neighborhood graph has trouble when distance between data points in different regions of the graph is different.

On the other hand, the k-nearest neighbor graph may connect a data point in a sparse cluster with a nearby dense cluster or it may break a cluster into several clusters if there are high density regions within a cluster that are separated from each other.

The mutual k-nearest neighbor graph has a property that it tends to connect points within regions of constant density but does not connect regions of different densities. Thus it does well with graphs of different scales but not with graphs with a mixture of multiple scales.

For Gaussian, generally a fully connected graph is used but the parameter σ plays the role of ϵ here by drastically reducing the weights of the edges between points that are far away from each other.

The author suggests to work with k-nearest neighbor graph as first choice as it is simple and comparatively robust to poor choice of parameters.

8.1.3 Parameters of the graph

Although there are theoretical results that may guide the choice of k or ϵ, very few are of practical use because most results hold in the limit for sample size $n \to \infty$.

For rules of thumb, as the author puts it, when working with k-nearest neighborhood graph, choose k such that resulting graph is connect or has significantly fewer connected components. e.g. for a large graph, start with k in order of $log(n)$

For mutual k-nearest neighbor graph, the k is significantly larger than that for k-nearest neighbor graph, but no such heuristic is available to choose the parameter k.

For ϵ-neighborhood graph, author suggests to choose ϵ as the length of the longest edge of minimal spanning tree of fully connected graph. There are several algorithms to determine the minimal spanning tree. However when outliers exist or when clusters are too far, this will choose very large values, thus connecting outliers with the rest of the data.

Finally, for fully connected graph, a scaling of similarity function can be used so that the points with similarity significantly larger than 0 are not too large or small. One can choose σ to be mean of distance to its k-th nearest neighbor or by minimal spanning tree.

Finding such rules with theoretical justification is considered an interesting important topic for future research.

8.2 Computing the Eigenvectors

Computing the k-eigenvectors of a large matrix by itself may be an expensive task. Fortunately, in practice, several edges in the graph are removed using k-nearest neighbor or ϵ-neighborhood graph. This results in a sparse matrix instead of a full matrix. Now, sparse eigensolvers like power method or Krylov-space methods can be used. These algorithms converge fast when the spectral gap $\gamma_k = |\lambda_k - \lambda_{k+1}|$ is large. There is a problem when the multiplicity of an eigenvector is more than 1. Also the numerical eigensolvers do not always converge to the cluster indicator vectors. Fortunately, all the vectors in the space spanned by the indicator vectors have to be of the form $u = \sum_{i=1}^{k} a\mathbf{1}_{A_i}$. i.e. they are piece-wise constant on the clusters. So the vectors returned by the eigensolver still encode the required information and can be used by k-means algorithm to compute the cluster.

8.3 Number of clusters

Choosing or detecting the number of clusters is a general problem for all the clustering algorithms and a variety of methods exist. Some statistical methods are based on log-likelihood of the data which can be treated in Bayesian or frequentist way. Other examples are methods based on ratio of within-cluster to between cluster similarity, gap statistic, etc. All these methods can be used in spectral clustering.

But additionally one tool specially designed for spectral clustering is eigengap heuristic. The goal is to choose k such that the eigenvalues $\lambda_1, \dots, \lambda_k$ are small but λ_{k+1} is relatively large. It has been shown that the cluster sizes are closely related to size of first eigenvalues. This technique works really well if the clusters are well differentiated. But as the cluster boundaries become fuzzy due to clusters being close or overlapping, the technique becomes less effective.

8.4 the k-means step

The algorithms studied in this paper all use k-means clustering. However, in principle any clustering technique can be used. Several authors have used Euclidian distance, hyperplane and other advanced post-processing techniques for eigenvectors to achieve the same goal.

8.5 Which graph Laplacian to be used?

The answer to this question largely depends on the data. The first aspect to look at is the degree distribution, which are the diagonal values of the degree matrix D. If most of the vertices have roughly the same degree, then all Laplacians would be equally effective. However due to various factors, some of which seen earlier in this paper, L_{rw} works better than L_{sym} or L.

8.5.1 Clustering objectives

In general, we have two objectives

1. Minimize between-cluster similarity, i.e. minimize $cut(A, \bar{A})$
2. Maximize within-cluster similarity, i.e. maximize $W(A, A)$ and $W(\bar{A}, \bar{A})$

Both Ncut and Ratiocut, by definition, directly implement the first criteria. However, second criteria is where the methods behave differently.

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

Thus within-cluster similarity $W(A, A)$ is maximized if $cut(A, \bar{A})$ is small and $vol(A)$ is large. This is implemented in Ncut only. We can see this even more clearly considering the MinimaxCut criterion defined as

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

Note that the denominator in Ncut is $vol(A) = cut(A, \bar{A}) + W(A, A)$. Thus, comparing Ncut and MinimaxCut, both are minimized when $cut(A, \bar{A})$ is minimized. In that case the denominator for Ncut and MinimaxCut will be very close. Thus Minimizing Ncut is the same operation as minimizing MinimaxCut, i.e. spectral clustering with eigenvectors of L_{rw}.

On the other hand, RatioCut focuses to maximize $|A|$ and $|\bar{A}|$, thus balancing the count of nodes. This is not related to the volume of cluster, which is based on the edge weights within the cluster. For this reason RatioCut is likely to cluster outliers together or with another cluster.

Thus we can see why clustering with L_{rw} is likely to yield superior results.

8.5.2 Consistency issues

Considering the statistical analysis of the Normalized and Unnormalized spectral clustering algorithms, we consider that the datapoints $x_1, \ldots, x_n$ to be sampled with a probability distribution P on some underlying dataspace χ. For both the normalized spectral clustering algorithms it can be proven that as we draw more and more data points, the spectral clustering results converge to to a partition of χ. This convergence implies that the eigenvalues and eigenvectors converge to those of U. Thus if we consider a diffusion process on the data space χ we can say that the partition induced by eigenvectors of U remains consistent (does not make a transition) most of the time. Thus, in real world situations, under mind conditions, all consistency statements about both L_{sym} and L_{rw} hold.

In contrast, the such consistency statements do not hold for unnormalized clustering. Although it is possible to show that such consistency problem can be avoided in situations where the first k eigenvalues are much smaller than the minimal degree of the graph, it is normally too restrictive in real world examples as it results in very few eigenvalues being useful (under the minimal degree constraint)

8.5.3 Which Laplacian?

All the evidence shown above and in earlier sections show that normalized spectral clustering has advantages over unnormalized spectral clustering. Also eigenvectors of L_{rw} are cluster indicators while those of L_{sym} are multiplied by $D^{1/2}$. Thus the final choice becomes L_{rw}.

Chapter 9

Closing remarks

Graph Laplacian and spectral clustering has been studied in many different fields by many researchers since 1973. Graph Laplacians have been applied to several use cases in addition to spectral clustering

An intuition often used graph Laplacian formally looks like a continuous Laplace operator considering the relationship between similarity and distance $w_{ij} = 1/d_{ij}^2$.

$$w_{ij}\big(f_i - f_j\big)^2 \approx \left(\frac{f_i - f_j}{d_{ij}}\right)^2$$

This looks like discrete version of the standard Laplace operator $\mathcal{L}$ on $\mathbb{R}^n$ which satisfies

$$\langle g, \mathcal{L}g \rangle = \int |\nabla g|^2 dx$$

This relation has been studied and shown to be precise. Apart from the application to partitioning problem,s graph Laplacians have been also studied and used for the connection between topology and properties of graphs and graph Laplacians.

Bibliography

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

[2] Helmut Lütkepohl. *Handbook of Matrices.* Wiley, 4th edition.

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

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

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

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