Spectral Clustering

Georgy Noarov

December 6, 2018

1 Introduction

Today's topic is Spectral Clustering. Clustering is a standard technique in data analysis, which is useful in many contexts, for example for preprocessing a given data set in order to facilitate feature extraction. Spectral Clustering is a particularly insightful approach to clustering which is based on spectral properties of the similarity graph of the data set, and on the idea of minimizing an appropriately defined notion of graph cut.

The outline of the lecture is as follows. In Section 2 we look at the goals of clustering in general, and specifically k-means clustering, which is a simple and ubiquitous way to cluster data when we are expecting clusters whose shape is not-too-complicated (e.g., convex). In Section 3, we introduce the fundamental notions of the Similarity Graph, the Graph Laplacian along with its variants (the Random Walk Laplacian and the Symmetric Laplacian), and two important types of Graph Cuts: the Ratio Cut, and the Normalized Cut. We consider their immediate properties relevant to the spectral approach to clustering. In particular, we show how to relax the discrete optimization problems corresponding to the minimization of Ratio Cuts and Normalized Cuts. Based on this, we derive the relationship between the second eigenvalue of appropriate Graph Laplacians and the minimum value of Ratio Cuts and Normalized Cuts. Section 3 also contains hints at how and why spectral considerations could be useful in clustering.

In Section 4, we define the Shi-Malik Normalized Cut Algorithm. It is one of the best known spectral clustering algorithms. Finally, in Section 5 we derive Cheeger's Inequality. It shows that the Normalized Cut Algorithm outputs clusters with close-to-optimal Normalized Cut. This can be viewed as a performance guarantee for the Shi-Malik Algorithm.

2 K-means Clustering

First, let us recall what goals we have in mind when we apply clustering to a data set. Naturally, the first goal is to find whether the data can be partitioned into some number of disjoint subsets, or *clusters*. The second objective, which is sometimes not kept firmly in mind, is also important: we want the points within each cluster to be as ”close together”, or ”similar”, as possible. Sparse clusters, after all, may not be meaningful and may just be a consequence of the fact that the number of clusters that we want to obtain on our data set is too small and should be increased. To summarize:

Objective 1 of Clustering Minimize inter-cluster relationships

Objective 2 of Clustering Maximize intra-cluster (within-cluster) relationships (that is to say, reduce intra-class variance)

Now let us talk briefly about what way of representing data is most amenable for clustering. Suppose we have our data set X and want to feed it to a clustering algorithm. Then, a simple idea is to embed our data set into the space $\mathbb{R}^d$ for some selected dimensionality d. Then, our data set becomes transformed into a set of points $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$. In this setting, the natural way to define clusters is geometric: roughly speaking, a subset of points belongs to the same cluster if they are close together. Proximity in this case is quite naturally measured in terms of the usual Euclidean distance: $dist_{i,j} = \|x_i - x_j\|$. Points are the more similar, the closer they are.

A famous algorithm coined by LLoyd but known simply as k-means clustering takes as input the geometric representation of our data set as described above, as well as the desired number of clusters k. Then, it does the following:

- Step 1: Initialize random centers $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.
- Step 2: Repeat the loop consisting of the following two substeps until convergence:
  - Step 2.1: Assign each point x_i to cluster j such that μ_j is the closest center to x_i.
  - Step 2.2: Recompute the centers $\mu_1, \ldots, \mu_k$ based on the current assignment of points to clusters.

2.1 Motivation behind Lloyd's Algorithm

At first glance, the algorithm seems to be doing something erratic, but in fact it is trying to optimize an underlying function quantifying how "good" any partition $S_1, \ldots, S_k$ of the set of points is, in terms of the two main Clustering objectives stated above.

Indeed, let us begin with the following intuitive measure of how good a partition is:

$$J(S_1, \ldots, S_k) := \frac{1}{2} \sum_{j=1}^{k} \frac{1}{|S_j|} \sum_{l,m \in S_j} \|x_l - x_m\|^2 \text{ for any partition } S_1, \ldots, S_k \text{ of the set of points.}$$

Here, the inner summation represents the intra-cluster variances. These variances are weighted by the cluster sizes $|S_j|$, which represents the penalty for making clusters too small (indeed, if we just pick small clusters, then their variances will be small not due to the fact that points there are meaningfully similar, but just because there are few of them: as a limiting case, consider a cluster of size 1: its variance is 0, but it is quite meaningless). Therefore, based on J, our clustering task is simple: the best clustering of the data set is given by the partition

$$(S_1^*, \ldots, S_k^*) = \arg \min_{S_1, \ldots, S_k} J(S_1, \ldots, S_k).$$

To see why this is exactly the function that Lloyd's Algorithm is implicitly working with, define

$$\mu_i = \frac{1}{|S_i|} \sum_{i \in S_i} x_i.$$

Then, for each cluster S_j,

$$\begin{aligned}
\sum_{l,m\in S_j} \|x_l - x_m\|^2 &= \sum_{l,m\in S_j} \left(\|x_l\|^2 + \|x_m\|^2 - 2\langle x_l, x_m\rangle\right) \\
&= \sum_{l\in S_j} \left(|S_j|\,\|x_l\|^2 - 2|S_j|\langle x_l, \mu_j\rangle + \sum_{m\in S_j} \|x_m\|^2\right) \\
&= 2|S_j| \left(\sum_{l\in S_j} \|x_l\|^2 - |S_j|\,\|\mu_j\|^2\right) \\
&= 2|S_j| \left(\sum_{l\in S_j} \|x_l\|^2 + |S_j|\,\|\mu_j\|^2 - 2|S_j|\,\|\mu_j\|^2\right) \\
&= 2|S_j| \left(\sum_{l\in S_j} \left(\|x_l\|^2 + \|\mu_j\|^2 - 2\langle x_l, \mu_j\rangle\right)\right) \\
&= 2|S_j| \sum_{l\in S_j} \|x_l - \mu_j\|^2 .
\end{aligned}$$

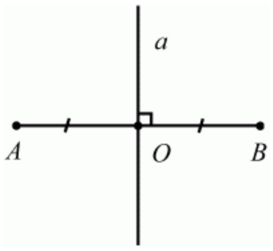
Therefore, in fact

$$J(S_1, \ldots, S_k) = \sum_{j=1}^{k} \sum_{l\in S_j} \|x_l - \mu_j\|^2$$

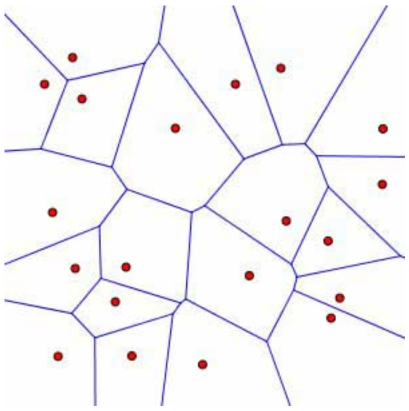
So we can see that in fact Lloyd's algorithm is trying to find clusters that would be optimal in terms of the above function, and thus in terms of the objective function J.

2.2 Some drawbacks of Lloyd's Algorithm

- It can be easily seen that k-means clustering is an algorithm that only finds *convex* clusters. This is because the boundaries between clusters are linear. Indeed, consider for simplicity the case of two clusters, and look at the means μ_1, μ_2. Then, Lloyd's algorithm employs classification based on which mean each point is closer to, which is a linear classifier. Indeed, the boundary is simply the normal to the segment connecting μ_1, μ_2 passing through its middle point:

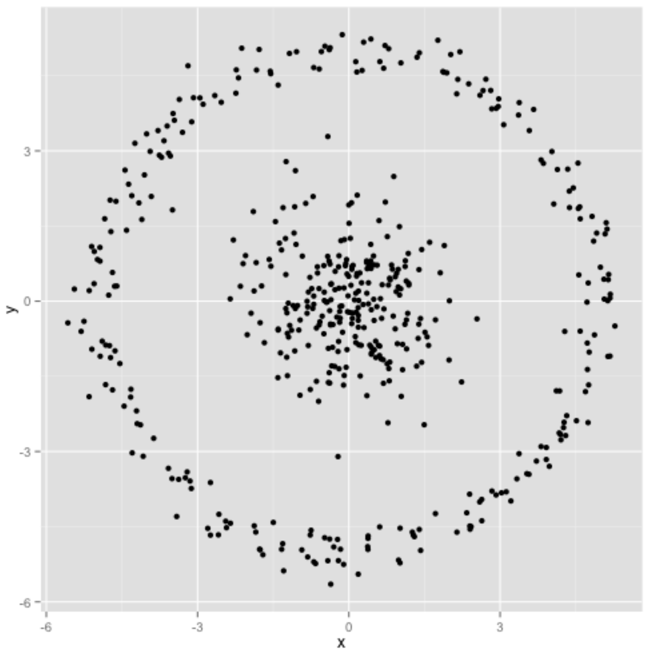


  Therefore, in the general case, the picture will look like this:



The red dots are the means, and the blue lines are the classification boundaries. Such structure is known as *Voronoi diagram*.

Clearly, such structure that the clusters have to conform to can be a poor choice when the real clusters in our data should look different (for example, nonconvex). Indeed, consider the following example of a dataset consisting of a spherical cluster surrounded by a concentric ring:



The two clusters here are obvious, but clearly there is no Voronoi diagram with 2 means that would split the image into these clusters. In fact, 2-means clustering will just split this dataset according to some line passing through the center of the dataset (the direction of that line depends on the initialization). On the homework, you will be asked to trace the execution of k-means clustering on a similar example.

- We also remark that there are no guarantees that Lloyd's Algorithm will converge to a global minimum of the objective function. Of course, the performance will still usually be satisfactory if the data set conforms to the assumption of convex, spherical-looking clusters, but just a thing to keep in mind.

3 The Similarity Graph and Related Notions

Let us take up the question we had in the previous section once again: how do we feed our data to a clustering algorithm so that the representation of the data set we supply to it is both relevant for clustering and compact? Unlike in the previous section, we don't have to embed our data in $\mathbb{R}^d$. There, Lloyd's algorithm was trying to find clusters that would minimize objective J, which is a function of the cluster sizes and the pairwise distances between points. So what is really essential is some notion of pairwise similarity. In general,

Definition 1 *A* similarity graph $G = (V, E)$ *of a dataset* $\{x_1, \dots, x_n\}$ *is a weighted undirected complete graph whose vertices are in bijective correspondence with the data points (vertex* i *corresponds to* x_i*), and the edge weight for every edge* (i, j) *is* $w_{i,j} = sim(x_i, x_j)$*, where* $sim(\cdot,\cdot) \geq 0$ *is a function representing the similarity between any two points in the data set: the larger* $sim(x_i, x_j)$*, the more similar* x_i *and* x_j *are.*

For example, the similarity function can be taken to be the inverse distance between points, in the example in Section 1, or in general it can be any conceivable function satisfying the above definition. A similarity graph can often be unweighted. For example the ε-neighborhood graph is defined so that points are connected via an edge iff they are at distance at most ε apart.

Thus, the answer to our question is the following: in many cases, clustering algorithms only need an appropriately constructed similarity graph of the to-be-clustered data set.

3.1 The Similarity Graph and its Laplacians

We define the following standard matrices associated with a similarity graph.

- The *degree matrix* $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $D_{ii} := d_i = \sum_{j\in V} w_{i,j}$, so that the main diagonal has all the degrees d_i of the vertices.
- The *adjacency matrix* $W \in \mathbb{R}^{n\times n}$ is a symmetric matrix with $W_{ii} := 0$ for all i and $W_{i,j} := w_{i,j}$ for all $i \neq j$. Therefore, the adjacency matrix just stores all edge weights of G.

Now we introduce the matrix associated with graph G that lies at the heart of an area in graph theory called *spectral graph theory*. It is called the *Laplacian matrix* and denoted by L (it is a function of graph G, like the above matrices). It is the matrix that induces a natural quadratic form (QF) on the functions $f : V \to \mathbb{R}$. The form is defined for such functions by

$$f^T L f = \sum_{i,j} w_{i,j}(f(i) - f(j))^2.$$

Here, $w_{i,j}$ are the edge weights of the similarity graph that L corresponds to. Note that we can interchangeably view f either as a function $V \to \mathbb{R}$ or as a vector $f \in \mathbb{R}^n$.

Definition 2 *The* Laplacian *of graph* G *is defined by* $L := D - W$.

Check yourself that indeed L induces the QF above. In some sense, that QF represents the variability of data set X. Let us consider the following toy problem with far-reaching consequences:

Embedding of a graph into the real line $\mathbb{R}$ (Hall '70) Consider a weighted undirected graph $G = (V, E)$. Find such embedding $f : V \to \mathbb{R}$ that the variance of the embedding,

$$\sum_{(i,j)\in E} (f(i) - f(j))^2$$

is minimized.

Since any embedding $f = const$ (as a vector, $f = \alpha\mathbf{1}$ for some real $\alpha > 0$ and $\mathbf{1} \in \mathbb{R}^n$ being the all-ones vector) would trivially minimize the objective, we also assume $f \perp \mathbf{1}$. We further assume $\|f\| = 1$, where the norm is the usual Euclidean norm of f viewed as a real vector. This is a way to prevent f from being arbitrarily close to the constant vector $\mathbf{0}$: if we did not make this restriction, then we would get a sequence of embeddings that make the objective arbitrarily close to 0.

Given these restrictions, the solution may sound surprising: the minimizer is the embedding $f = \phi_2(L)$, where $\phi_2(L)$ is the second eigenvector of L, the eigenvector corresponding to the second smallest eigenvalue λ_2. The minimum value of the objective is λ_2.

Proof. Note that L is symmetric, hence Hermitian, so it has n real positive eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, and one can choose the corresponding eigenvectors $\phi_1, \ldots, \phi_n$ ($L\phi_i = \lambda_i\phi_i$) so that they form an orthonormal eigenbasis. We tacitly wrote $\lambda_1 = 0$, since $L\mathbf{1} = 0$ and thus 0 is always its eigenvalue.

Write any f satisfying $f \perp \mathbf{1}$ and $\|f\| = 1$ in the orthonormal eigenbasis of L as $f = \sum_{i=1}^n \alpha_i\phi_i$. Then since $f \perp \mathbf{1}$, and we have just observed that $\phi_1 = \mathbf{1}$, we can rewrite $f = \sum_{i=2}^n \alpha_i\phi_i$. Then,

$$f^T Lf = \sum_{l=1}^{n} \lambda_l \alpha_l^2 \geq \lambda_2 \sum_{l=2}^{n} \alpha_l^2 = \lambda_2 \|f\|^2 = \lambda_2.$$

Thus, the minimum of the objective cannot be smaller than λ_2. Choose $\alpha_1 = 1$, and $\alpha_1 = \alpha_3 = \ldots = \alpha_n = 0$, and the vector $f = \phi_2$ will satisfy $f^T Lf = \lambda_2$, hence ϕ_2 is the minimizer.

■

Other Laplacians Two other versions of a graph Laplacian with similar properties are the following:

Definition 3 *The* symmetric *Laplacian is defined as $L_{sym} := D^{-1/2}LD^{-1/2}$. The* random walk *Laplacian is defined as $L_{rw} := D^{-1}L$.*

Their usefulness will become clear in the future, but as a preliminary note, the spectral properties of these Laplacians are very similar to those of the ordinary Laplacian L.

Important Property of Laplacians Come back here when you have seen the Shi-Malik algorithm in Section 4. This result provides intuition as to why spectral clustering works: in the trivial case of k disconnected clusters, the algorithm will return exactly these clusters.

Theorem 4 *Let G be a similarity graph. Then the multiplicity of eigenvalue 0 of L equals the number of connected components of G (we interpret 0-weight edges as "disconnected"). The eigenspace of 0 is spanned by the indicator vectors $1A_1, \ldots, 1_{A_k}$ of those connected components.*

Proof. Say the graph is connected. Then let f be an eigenfunction with eigenvalue 0. We have $0 = f^t Lf = \sum_{i,j} w_{i,j}(f_i - f_j)^2$. Hence $f = const$ for all vertices connected by a path, and thus it is constant on the entire graph.

Now say there are k connected components. Then W and thus $L = D - W$ will have block form, after a reordering:
$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{bmatrix}$. Thus each L_i is a proper graph Laplacian on its own. As for any diagonal block matrix the union of spectra of diagonal blocks L_i is the spectrum of L, and eigenvectors of L are eigenvectors of L_i's filled with zeros at the positions of other blocks, we have reduced our claim to each of the L_i's, which concludes the proof. ■

3.2 The Normalized Cut and the Ratio Cut

When we speak of successful clustering, we mean clusters that have strong intra-cluster association and weak inter-cluster association. Both these properties can be represented through the properties of the similarity graph. Specifically, we should find helpful the following notions:

Definition 5 *The* cut *associated with any subset S of vertices of graph G is defined as the total weight of all inter-cluster edges corresponding to the partition of V into S and its complement $\bar{S}$:*

$$cut(S) := \sum_{i \in S, j \in \bar{S}} w_{i,j}.$$

Definition 6 *The* volume *of any subset S of vertices of graph G is defined as the sum of weights of all edges whose both ends are contained within S:*

$$vol(S) := \sum_{i,j \in S} w_{i,j}.$$

Definition 7 *The* association *between any two disjoint subsets S and T of vertices of graph G is defined as the sum of weights of all edges connecting S and T:*

$$assoc(S) := \sum_{i \in S, j \in T} w_{i,j}.$$

Now, let us define two types of "cuts" that quantify how good a partition of a graph into two clusters is.

Definition 8 *The* ratio cut *corresponding to any partition $S, \bar{S}$ of the vertices V of graph G is defined to be*

$$Rcut(S) := \frac{cut(S)}{|S|} + \frac{cut(\bar{S})}{|\bar{S}|}.$$

The normalized cut *corresponding to any partition $S, \bar{S}$ of the vertices V of graph G is defined to be*

$$Ncut(S) := \frac{cut(S)}{vol(S)} + \frac{cut(\bar{S})}{vol(\bar{S})}.$$

Let us note that by minimizing the normalized or the ratio cut, we would aim at decreasing the value of $cut(S)$, which represents inter-cluster relationship. At the same time, minimizing $Ncut$ would also aim at maximizing $vol(S)$ and $vol(\bar{S})$. Note that $assoc(S, S) = vol(S) - cut(S)$, and therefore since minimizing $Ncut$ strives to minimize $cut(S)$ and maximize $vol(S)$, then we would be maximizing $assoc(S, S)$, which stands for intra-cluster relationships within cluster S. Thus, the declared goals of clustering would be accomplished by minimizing Ncut and, to a certain extent, Rcut.

It can also be shown that minimizing $Ncut$ leads to clustering that minimizes the probability of a random walker, who is walking on the clusters and jumps according to the strenghts of pairwise intercluster relationships, to jump from one cluster to another.

Another property of Ncut is that if we assume, in a statistical setting, that our data comes from common distribution P and is iid., then if we let the number of points $n \to \infty$, minimizing $Ncut$ would produce clusterings that converge, in a natural sense, to a limit clustering consistent with the density of the underlying distribution. Rcut almost never has this convergence property.

In fact, in most settings, using Ncut is preferable to Rcut, because it has more nice properties than Rcut.

3.3 Minimization of Ncuts and Rcuts and its Relaxation

The central question now becomes: can we actually efficiently find set $S \subset V$ that minimizes, say, $Rcut$? Note that there are $2^{|V|}$ possibilities to check, and thus unless there is a very clever algorithm that could search this exponentially big space in poly-time, we need to think of a different approach to minimizing $Rcut$. In fact, it turns out that this minimization problem is NP-hard. Therefore, we utilize the following trick: rather than handle this difficult discrete optimization problem, we extend the search space to be continuous.

First, let us define the discrete space corresponding to this problem as the set of vectors of length $n = |V|$, each of which stands for one of the subsets $S \subset V$ and has only two specific values as its entries. Namely, for each $S \subset V$,

$$f_S(i) = \begin{cases} \sqrt{\frac{|\bar{S}|}{|V||S|}} & , i \in S \\ -\sqrt{\frac{|S|}{|V||\bar{S}|}} & , i \in \bar{S} \end{cases}.$$

A beautiful property of each f_S is that the natural quadratic form of the Laplacian is equal to $Rcut(S)$ when evaluated at that vector. Indeed,

$$\begin{aligned} f^T L f_S &= \frac{1}{2}\sum_{i,j} w_{i,j}(f_S(i) - f_S(j))^2 \\ &= \sum_{i \in S, j \in \bar{S}} w_{i,j}\left(\sqrt{\frac{|\bar{S}|}{|V||S|}} + \sqrt{\frac{|S|}{|V||\bar{S}|}}\right)^2 \\ &= \frac{1}{|V|}\sum_{i \in S, j \in \bar{S}} w_{i,j}\left(\frac{|\bar{S}|}{|S|} + \frac{|S|}{|\bar{S}|} + 2\right) \\ &= \frac{1}{|V|}\sum_{i \in S, j \in \bar{S}} \left(\frac{|\bar{S}| + |S|}{|S|} + \frac{|S| + |\bar{S}|}{|\bar{S}|}\right) = \frac{cut(S)}{|S|} + \frac{cut(\bar{S})}{|\bar{S}|} \\ &= Rcut(S). \end{aligned}$$

So when we want to minimize $Rcut(S)$, we just need to look at each of these vectors and compute the Laplacian QF on them, and pick the subset $S^* \subset V$ corresponding to the argmin vector. But note that we still have not reduced the size of the search space, just reformatted it. To proceed, note the following additional properties of f_S.

- First, all these vectors are normalized:

$$\|f_S\|^2 = \sum f_S(i)^2 = \frac{1}{|V|}\left(\sum_{i\in S}\frac{|\bar{S}|}{|S|} + \sum_{i\in\bar{S}}\frac{|S|}{|\bar{S}|}\right) = \frac{1}{|V|}(|\bar{S}| + |S|) = 1.$$

- Second, these vectors are all orthogonal to the all ones vector:

$$f_S^T\mathbf{1} = \sum_{i\in V} f_S(i) = \sum_{i\in S}\sqrt{\frac{|\bar{S}|}{|V||S|}} - \sum_{i\in\bar{S}}\sqrt{\frac{|S|}{|V||\bar{S}|}} = \sqrt{\frac{|S||\bar{S}|}{|V|}} - \sqrt{\frac{|\bar{S}||S|}{|V|}} = 0.$$

Now let us relax the above optimization problem, that is, we let the vectors f range not just within the space of $\{f_S|S \subset V\}$, but over the entire space $\mathbb{R}^{|V|}$ of real-valued vectors of length $|V|$. We also make these vectors subject to the two restrictions identified above on the f_S vectors, but no further restrictions. This way, the vectors f_S will be a part of the new search space! Explicitly, consider the following optimization problem *over the real-valued vectors of length* $|V|$:

$$\begin{aligned}&\text{minimize } f^T L f\\ &\text{subject to } \|f\| = 1 \text{ and } f\mathbf{1} = 0.\end{aligned}$$

As proved above, this problem is solved by $\phi_2(L)$, the second eigenvector of the Laplacian, and the value achieved at that vector is $\lambda_2(L)$. Based on the above remark that the discrete search space is subsumed within the new search space, we have that the objective value achieved on the new set is at most that achieved on the old set:

$$\min_{S\subset V} Rcut(S) \geq \lambda_2(L).$$

It turns out that a similar relaxation approach makes minimizing $Ncut(S)$ easy: namely, the relaxed problem over the real vectors f of length $|V|$ in that case is explicitly

$$\begin{aligned}&\text{minimize } g^T L_{sym} g\\ &\text{subject to } \|g\| = 1 \text{ and } g^T D^{1/2}\mathbf{1} = 0,\\ &\text{where } g = D^{1/2} f.\end{aligned}$$

This is a homework problem.

Furthermore, notice that $L_{sym}D^{1/2}\mathbf{1} = D^{-1/2}L\mathbf{1} = 0$, which means that $D^{1/2}\mathbf{1}$ is an eigenvector of L_{sym} with eigenvalue 0. Thus, in the same way as for the preceding optimization problem, we have that the solution vector g^* is the second eigenvector $\phi_2(L_{sym})$. Therefore, the minimizer $f^* = D^{-1/2}g^* = D^{-1/2}\phi_2(L_{sym}) = \phi_2(L_{rw})$, which is the second eigenvector of L_{rw}, the random walk Laplacian. The corresponding value of the objective function is then $\lambda_2(L_{rw})$, the second eigenvalue of the random walk Laplacian (and of the symmetric Laplacian).

3.3.1 Extension to multiple clusters

We remark that both $Ncut$ and $Rcut$ can be easily defined for partitions of the vertex set of graph G into more than 2 subsets (above, we had the partition of V into S and $\bar{S}$.) In precise terms, we set

$$Ncut(S_1, \dots, S_k) = \sum_{j=1}^{k} \frac{cut(S_k)}{vol(S_k)}$$

for any partition $S_1, \dots, S_k$ of V into k clusters.

In order to extend the logic behind relaxation programs from above, we set k indicator vectors h_{S_i} as follows:

$$h_{S_i}(j) := \begin{cases} \frac{1}{\sqrt{vol(S_k)}} & , j \in S_i \\ 0 & , j \notin S_i \end{cases}$$

Now let us combine these indicators into a single matrix:

$$H = [h_{S_1} \dots h_{S_k}].$$

From the last problem on the Homework, we know the following: that

$$Ncut(S_1, \dots, S_k) = Trace(H^T L H),$$

and

$$H^T D H = I_k.$$

These properties are in direct parallel with the restrictions in the relaxed $Ncut$ problem.

Therefore, in order to figure out the spectral relaxation algorithm for k cuts, we need to minimize $Tr(H^T L H)$ subject to $H^T D H = I_k$. This is equivalent to minimizing $Tr(G^T L_{sym} G)$ subject to $G^T G = I_k$, if we let $G = D^{1/2} H$.

Theorem 9 *In the relaxation above, the minimizer is $G = [g_1 g_2 \dots g_k]$, where $g_1, \dots, g_k$ are the first k eigenvectors of L_{sym}, that is those eigenvectors corresponding to $\lambda_1(L_{sym}), \dots, \lambda_k(L_{sym})$.*

The minimizer H of the original relaxation problem is thus $H = [h_1 h_2 \dots h_k]$, where $h_i = D^{-1/2} g_i$ are the first k eigenvectors of L_{rw}.

Proof. The first statement is clear from the homework exercise. The second statement is because $L_{rw} h_i = D^{-1/2} L_{sym} D^{1/2} (D^{-1/2} g_i) = D^{-1/2} L_{sym} g_i = \lambda_i D^{-1/2} g_i = \lambda_i h_i$. ∎

4 A Successful Spectral Clustering Algorithm

We now present the following important algorithm due to Shi-Malik that belongs to the class of Spectral Clustering algorithms:

- INPUT: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters.
- Steps:
  - Step 1: Construct similarity graph G, compute its normalized Laplacian L_{rw}
  - Step 2: Compute the first k eigenvectors $p_1, \dots, p_k$ for the Laplacian L_{rw}
  - Step 3: Let $P \in \mathbb{R}^{n \times k}$ be the matrix $[p_1 \dots p_k]$. For $i = 1 \dots n$, let $y_i \in \mathbb{R}^k$ be the i-th row of P.
  - Step 4: Cluster $\{y_i\}_{i=1}^{n}$ in $\mathbb{R}^k$ using Lloyd's algorithm for k-means clustering, and thus obtain clusters $S_1, \dots, S_k$
- OUTPUT: Clusters $C_1, \dots, C_k$ such that $C_i = \{v | y_v \in S_i\}$.

5 Cheeger's Inequality

Cheeger's Inequality will provide guarantees on the performance of the Normalized Cut Algorithm presented above. Recall that we transitioned from the discrete minimization problem for Ncut to the corresponding relaxation, but there are still two questions hanging in the air:

- We know the solution to the relaxed problem, and it is the 2nd eigenvector ϕ_2 of L_{rw}. The objective function value for this vector will then be $\lambda_2(L_{sym})$. However, how close is $\lambda_2(L_{sym})$ to the minimum value $\min_{S \subset V} Ncut(S)$ that we are seeking?
- How do we threshold the obtained vector ϕ_2 to convert it from a real vector to a vector of 0s and 1s, where 0 denotes belonging to S and 1 stands for belonging to $\bar{S}$? Is thresholding even the right approach here?

Cheeger Inequality provides answers to these questions in particular.

Definition 10 *The* Cheeger constant *of graph G is defined to be*

$$h_G := \min_{S \subset V} \frac{cut(S)}{\min\{vol(S), vol(\bar{S})\}}.$$

Theorem 11 (Cheeger's Inequality)

$$\frac{h_G^2}{2} \leq \lambda_2(L_{sym}) \leq 2h_G$$

The proof of the right inequality is very easy. Recall that for any set S,

$$Ncut(S) = cut(S)\left(1/vol(S) + 1/vol(\bar{S})\right).$$

Thus, $Ncut(S) \leq 2\frac{cut(S)}{\min\{vol(S), vol(\bar{S})\}} \leq 2h_G$. Furthermore, we know from the relaxed $Ncut$ minimization problem that $\lambda_2(L_{sym}) \leq \min_{S \subset V} Ncut(S)$. Therefore, $\lambda_2(L_{sym}) \leq 2h_G$, Qed.

For the proof of the left inequality, it is quite involved and long, so below will follow the highlights, and for the full proof please consult Lecture Notes 14 by Prof. Singer.

Step 1: Order data points, embed them into the real line, partition in two clusters First, one considers function g which provides the solution to the relaxed Ncut minimization problem. In other words, g is the minimizer of the Rayleigh-Ritz quotient $R(g) = \frac{g^T Lg}{g^T Dg}$. Then reorder the points x_i so that the vertices v_i of G satisfy $g(v_1) \geq \ldots \geq g(v_n)$, and consider the sets $S_i = \{v_1, \ldots, v_i\}$. Then, by definition of Cheeger's constant, it suffices to prove that the ratio $h_S := \frac{cut(S)}{\min\{vol(S), vol(\bar{S})\}}$ evaluated on at least one of the S_i's (say S_j), satisfies $h_{S_j}^2/2 \leq \lambda_2$.

Intuition: Once we have proved that, we will have automatically proved the following. For $k = 2$, if we take the minimizer g of the relaxed problem, then the optimal partition among those obtained by thresholding it (by using a single real threshold r) is within a quadratic factor of the optimum. Indeed, each S_i corresponds to some way to set a threshold r on the entries of g, so that all i such that entry $g(i) \leq r$ are assigned to S, and the rest - to $\bar{S}$. And then, we should have $\min_i h_{S_i} \leq \sqrt{2\lambda_2} \leq \sqrt{4h_G}$.

Now, we let r be the threshold index that will intuitively give a good value of $Ncut(S_r)$. Formally, set r to the largest integer such that $vol(S_r) \leq \frac{1}{2} vol(G) \leq vol(S_{r+1})$.

Intuition: This r should work well, since it partitions the vertices in a "balanced" fashion with respect to the volumes of the two clusters.

Step 2: Inspect the partition, split $g - g(v_r)$ into positive and negative part, and relate $R(g)$ to h_{S_r} The optimal Rayleigh quotient is (recall the QF associated with L)

$$R(g) = \frac{\sum_{i,j} w_{u,v}(g(u) - g(v))^2}{\sum_u d_u g(u)^2}.$$

Consider our embedding, split by the index r into the "left" and "right" parts. We want to show that, if we "threshold" our vector g so that all points on the "left" are collapsed into the same value a, and the points on the right are all collapsed into the same value b, then the vector $g_{collapse}$ formed this way has Rayleigh quotient $R(g_{collapse})$ that is not too far from the optimal quotient $R(g)$.

Intuition: Once we have collapsed the graph like this, all the differences $g_{collapse}(u) - g_{collapse}(v)$ for u, v that were on the same side ("left" or "right"), will have disappeared from the numerator of the Rayleigh quotient. Therefore, it is hard to relate $R(g_{collapse})$ and $R(g)$ since most terms in the numerator will have vanished. Therefore, we will first collapse just the left vertices, forming the corresponding vector g_-, and look at how far $R(g_-)$ is from $R(g)$, and then we will collapse just the right vertices, forming the corresponding vector g_+, and look at how far $R(g_+)$ is from $R(g)$. Then, we will relate $R(g_{collapse})$ to $R(g)$ through $R(g_-)$ and $R(g_+)$. It becomes quite clear by looking at the definitions of g_- and g_+ that their Rayleigh quotients will be related to $R(g_{collapse})$ in the following way: for some real numbers A, B, C, D, $R(g_+) = \frac{A}{C}$, $R(g_-) = \frac{B}{D}$, and $R(g_{collapse}) = \frac{A+B}{C+D}$. Since $\frac{A+B}{C+D} \leq \min\left(\frac{A}{C}, \frac{B}{D}\right)$, this relation is quite natural indeed.

The right way to define these vectors g_+ and g_- is: $g_+(v) = g(v) - g(v_r)$ if $g(v) \geq g(v_r)$ and $g_+(v) = 0$ otherwise. And $g_-(v) = -g(v) + g(v_r)$ if $g(v) < g(v_r)$ and $g_-(v) = 0$ otherwise. Thus, g_+ and g_- are simply the positive and negative parts of $g - g(v_r)$.

For the rest of the proof, refer to Prof Singer's notes.

6 Bibliography and Where to Refer for Details

6.1 List of References and their Corresponding Sections in the Notes

1 Prof. Singer's Notes for MAT 586, Lecture 13

- **Corresponding Sections:** Sections 2-4

2 Prof. Singer's Notes for MAT 586, Lecture 14

- **Corresponding Sections:** Section 5