1 A Tutorial on Spectral Clustering Notes

1.1 Understanding Normalized Cuts

We define a normalized cut of k partitions as $\text{NCut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{1}{Vol(A_i)} \cdot W(A_i, \overline{A_i})$ where $W(A_i, \overline{A_i}) = \sum_{a \in A_i, b \notin A_i} w_{ab}$ (the sum of the total edge weight going across the two sets). We wish k clusters that minimizes the normalized cut. On a graph with edges representing similarity, minimizing Ncut would pick out clusters that prefer similar vertices within each cluster (maximizing $vol(A_i)$ minimizes $\frac{1}{vol(A_i)}$) and dissimilar vertices across different clusters.

We use the idea of indicators to denote if a vertex i belongs to cluster j where

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}} & i \in A_j \\ 0 & i \in \overline{A_j} \end{cases}.$$

Now, let H be the k column vectors ($h_j = (h_{1,j}, ..., h_{n,j})$), where each column corresponds to a different cluster j:

$$H = \begin{bmatrix} | & | & & | \\ h_1 & h_2 & \cdots & h_k \\ | & | & & | \end{bmatrix}$$

Recall that $vol(A_j) = \sum_{i \in A_j} d_i$. Then,

$$h_j' D h_j = \sum_{i \in A_j} \frac{d_i}{vol(A_j)} = 1 \text{ and } h_i' L h_j = 0 \text{ for all i!=j, so } H'DH = I.$$

$$\begin{aligned} h_j' L h_j &= \frac{1}{2} \sum_{u,v \in A} w_{uv}(h_{uj} - h_{vj})^2 \\ &= \sum_{u \in A_j, v \in \overline{A_j}} w_{uv}(h_{uj} - h_{vj})^2 \\ &= \sum_{u \in A_j, v \in \overline{A_j}} w_{uv}(\frac{1}{Vol(A_j)}) \\ &= \frac{Cut(A_j, \overline{A_j})}{Vol(A_j)} \\ &= (H'LH)_{j,j} \end{aligned}$$

$$NCut(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{Cut(A_i, \overline{A_i})}{Vol(A_i)} = tr(H'LH).$$

We see that minimizing the normalized cut is the same as minimizing $tr(H'LH)$. However, as is, with the restrictions on what the columns can take on, this problem is NP-complete and computationally intractable for large graphs. Relaxing this problem by removing the indicator constraint on the column vectors reduces this problem to an eigenvector problem. The problem is then

$$\begin{aligned} \min_{H} \quad & tr(H'LH) \\ \text{s.t.} \quad & H'DH = I. \end{aligned}$$

Suppose we let $T = D^{\frac{1}{2}}H$, so $tr(H'LH) = tr(T'D^{-\frac{1}{2}}LD^{-\frac{1}{2}}T) = tr(T'L_{sym}T)$

$$\begin{aligned} \min_{T} \quad & tr(T'L_{sym}T) \\ \text{s.t.} \quad & T'T = I. \end{aligned}$$

The Rayleigh-Ritz Theorem tells us that the eigenvectors corresponding to the k smallest eigenvalues of L_{sym} solves the above minimization problem. The matrix H that solves the original minimization problem is $H = D^{-\frac{1}{2}}T$ which coincidentally correspond to the smallest k eigenvectors of H. With this in mind, Shi and Malik came up with the following algorithm:

Algorithm 1: [Shi and Malik] Normalized Spectral Clustering

Result: Return clusters $A_1, \cdots, A_k$

Given a similarity graph S $\in R^{n \times n}$, k clusters

1. Construct a similarity graph according to k-nearest neighbors, ϵ-neighborhood graph, or fully connected graph.
2. Create $L_{rw} = D^{-1}L$ where L is the Laplacian of the similarity graph.
3. Create a matrix H for L_{rw} with the k eigenvectors corresponding to the smallest k eigenvalues as the columns.
4. Run k-means on the rows of H to cluster the vertices.

One thing that isn't immediately clear is why looking at the distance between the rows of H tells us anything meaningful. The paper touches on commute distance which is the expected distance travelled before returning to the original point. More exploration on this later. Although not a satisfying answer, a smart relaxation of this problem gives us an answer close to the unrelaxed version. As mentioned in the paper, there is no guarantee this algorithm gives us a good clustering.

1.2 Random Walks

Algorithm 1 can be understood through the lens of a random walk. The matrix $D^{-1}L$ is a transition matrix with the rows being the transition probabilities for the corresponding matrix. Similar to the cut notion of a good clustering (low edge weights/similarity between clusters and high similarity within a cluster), a good clustering in terms of random walks refers to a partition of the graph such that a random walk stays within the same cluster for as long as possible and has low probability of jumping into other clusters.

1.3 Perturbation theory

If we consider a graph with k connected components, we know for a fact that there are k 0-eigenvalues with eigenvectors that correspond to indicator functions for each component. Now, if we add low edges weights between the disconnected components, the eigenvectors are very close to the ideal indicator vectors. This allows the k-means algorithm to separate the groups from each other.

1.4 Different similarity graphs and heuristics for choosing constants

Typically, we use the Gaussian similarity function, $s(x_i, x_j) = exp(-||x_i - x_j||^2/2\sigma^2)$.

1. k nearest neighbor graph: can break high density regions into disconnected components if they are reasonably far from each other, k is hard to determine
2. mutual k nearest neighbor graph: good at detecting clusters of different densities, k is hard to determine

3. ϵ-neighborhood graph: not good for data of different densities and difficult to choose ϵ
4. fully connected graph: not a sparse graph

To guarantee a fully connected graph, we choose ϵ to be the shortest edge of the minimum spanning tree. Use caution when the dataset contains outliers and when there are dense clusters far apart from each other because ϵ will be chosen to be too large to be meaningful.

If using a fully connected graph, the similarity function should mimic the properties of an ϵ-neighborhood graph and k-nearest neighborhood graph. To achieve this, we can choose σ to be the mean distance of a point to its k-nearest neighbor or the minimum edge weight of a minimum spanning tree.

1.5 Which laplacian should I use?

If the degree distribution of the similarity graph is roughly the same, it doesn't matter which Laplacian is used. On the other hand, if the degrees differ, normalized spectral cluster is better because the problem it reduces to– normalized spectral clustering which reduces to $NCut(\cdot)$ helps find a partition that maximizes the within-cluster similarities (depends on $Vol(\cdot)$) and minimizes the between-cluster similarity while unnormalized spectral clustering reduces to $Cut(\cdot)$ and just does the latter (depends on $|\cdot|$). Both the normalized laplacians converge to those of U but the unnormalized laplacian can fail to converge.

L_{rw} over L_{sym} because the eigenvector for L_{sym} is multiplied by $D^{1/2}$ and doesn't have computational advantages.

1.6 Thoughts and Future Steps

It's mentioned that the Euclidean distances between the $y_i's$ can be interpreted as building clusters of the vertices in the graph based off the commute distance. I don't get any intuition from this statement and would like to delve deeper to figure this out. Apparently commute distance is supposedly useless as the number of vertices approaches infinite; does that imply that the algorithms break down as we have more vertices? Perhaps delving deeper into the random walk interpretation and potentially studying Markov Chains and stochastic processes might help.

Perturbation theory seems like a really rich field and there is a lot to be learned. The angle stuff doesn't make much sense currently.

Really interested in why our eigenvalues dips below 0. Potentially due to solving $Lv = \lambda Dv$. Computationally, would it be better to solve for v as in $D^{-1/2}LD^{-1/2}u = \lambda u$ then compute $v = D^{-1/2}u$ and why?

The reason behind why the Laplacian matrix is called a Laplacian seems really important. An opportunity for me to learn what a Laplacian is... and about PDEs?

Could be fun to use spectral graph clustering on the handwritten digits. To classify new values, we just see how it fits into the graph. We know how many clusters there will be so maybe we can explore other properties.

Figure out how to use a bibliography, do footnotes, etc. Feels wrong not to cite the paper we read.