Spectral Clustering

This is a lecture scribe from ENEE759G: Unsupervised Learning-Fall 2019

1 Starting Point

For a set of n data points $x_1, x_2, ..., x_n \in \mathbb{R}^d$, we can construct a weighted graph $G = (V, E)$ where each vertex represents a data point and for every edge, there is a weight W_{ij} which measures the similarity between v_i and v_j. A commonly used similarity measurement is Gaussian kernel as shown in equation 1 which is a non-linear function of Euclidean distance. For this similarity measurement, the smaller the weight is, the more similar are the two data points.

$$W_{ij} = exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \tag{1}$$

To this point, the task of clustering can be described as follows: Partition a graph into natural groups so that the nodes in the same cluster are more "similar" to each other than those in different clusters[1]. To measure the quality of the clustering, we need a objective function to minimize or maximize. Thus, *cut* is introduced here as a naive objective function.

2 Cut

First, we will give the definition of *cut*. To do so, we first let V_1 be a subset of V and $\bar{V_1}$ be the complement of V_1. Then, we define

$$W(V_1, \bar{V_1}) = \sum_{i \in V_1, j \in \bar{V_1}} w_{ij} \tag{2}$$

which is the sum of the weights that connects the subset V_1 and $\bar{V_1}$. Now the *cut* can be defined as

$$Cut(V_1, \bar{V_1}) = \frac{1}{2} W(V_1, \bar{V_1}) \tag{3}$$

2.1 Minimal Cut

The *cut* alone can be used as the objective function. Then the take of clustering can be defined as finding the clusters that can minimize the value of the *cut* defined in equation 3. More formally

$$\text{Find } V_1, V_2, ..., V_k \text{ such that } \sum_{l=1}^{k} Cut(V_l, \bar{V_l}) \text{ is minimized}$$

We assume that the value of k is known for now.
In figure 1 is an simple example. If this is the result of minimal cut, we can say that the sum of w_1, w_2, w_3 is minimum.

But the minimal cut have some drawbacks such as unbalanced partition. Let the value of w_4 be very small, then the minimum cut will treat v itself as a cluster and all the other vertices as another cluster which is obviously not optimal. To address this problem, we introduce two modified version of minimal cut.

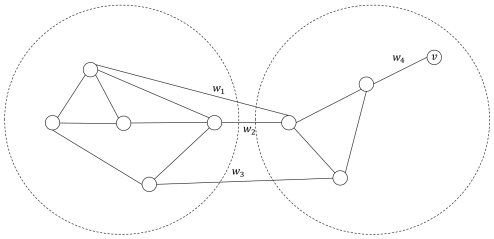

Figure 1: The result of minimum cut

2.2 Ratio Cut

Instead of using vanilla cut as the object function, We can use $RatioCut$ as objective function. The $RatioCut$ is defined as

$$RatioCut = \sum_{l=1}^{k} \frac{Cut(V_l, \bar{V}_l)}{|V_l|}$$

where $|V_l|$ is the number of vertices in A_l. The goal becomes finding $V_1, V_2, ..., V_k$ such that the $RatioCut$ is minimized. By adding the size of V_l to the objective function, preference will be given to the partitions with more balanced clusters.

2.3 Normalized Cut

Another version of Cut is $NormalizedCut$ (hereafter $NCut$). Instead of used the size of each cluster, the sum of the weights of all edges attached to vertices in A is used in $NCut$. Thus, the $NCut$ is defined as[2]

$$NCut = \sum_{l=1}^{k} \frac{Cut(V_l, \bar{V}_l)}{Vol(V_l)}$$

$$\text{where } Vol(V_l) = \sum_{i \in V_l} \left(\sum_{j=1}^{n} w_{ij} \right)$$

The goal here is to find $V_1, V_2, ..., V_k$ such that the $NCut$ is minimized.

3 Laplacian Matrix

An straightforward method the solve the problems mentioned before is simply try all the possibilities. This is not not practical, so we need some efficient algorithms. One of them is using Laplacian matrix. Given a graph with n vertices, its Laplacian matrix $L_{N \times N}$ is defined as

$$L = D - W \tag{4}$$

where D is an N by N diagonal matrix defined as

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}$$

$$\text{where } d_l = \sum_{j=1}^{n} w_{lj}$$

W is the adjacency matrix[1] defined as

$$W = (w_{ij})$$
$$w_{ij} = exp(\frac{-||x_i - x_j||^2}{2\sigma^2})$$

There are two important facts about the Laplacian matrix:

- for any $x \in \mathbb{R}^n$
$$x^T L x = \frac{1}{2} \sum_{i,j} w_{ij}(x_i - x_j)^2 \tag{5}$$

- Let λ_i be the $i-th$ eigenvalue of Laplacian matrix, then we have
$$0 = \lambda_0 \geq \lambda_1 \geq ... \geq \lambda_n$$
The eigenvector corresponding to $\lambda_0 = 0$ is $v_0 = [1, 1, ..., 1]^T$
Proof: $Lv_0 = (D - W) \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 0v_0$

Theorem 1. *G has P connected components if and only if $\lambda = 0$ is an eigenvalue of L with multiplicity P.*

Proof:

- =>: Let G have P connected components and we can denote them as $(V_1, E_1), (V_2, E_2), ..., (V_p, E_p)$. For each connected components, we define an indicator vector which is a column vector with size n. The i-th elements of indicator vector is 1 if the i-th vertex is in this connected component and 0 otherwise. We can Immediately tell from the definition that all the indicator vectors are linear independent.
We reorder the vertices in the graph such that the vertices in the first connected components comes the first and then the elements in the second connected components and so on. We also modify the indicator vectors correspondingly. After the reordering, the L matrix will be as follows

[1]The adjacency matrix A for a graph G is a $N \times N$ square matrix where N is the number of vertices in G. Its element A_{ij} is one (here the value should be the value of similarity measurement) when there is an edge from vertex i to vertex j, and zero when there is no edge.

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_p \end{bmatrix}$$

where each L_i is the Laplacian matrix corresponding to i-th connected components. And the i-th indicator vector will be as follows

$$e_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Since e_i is the eigenvector corresponding to the $\lambda = 0$ for i-th connected component. When multiplying L to each e_i, we will get

$$Le_i = L_i e_i = 0 = 0e_i$$

There are total of P connected components. Thus, we can conclude that $\lambda = 0$ is an eigenvalue of L with multiplicity P.

- <=: Let $\lambda = 0$ be the eigenvalue with multiplicity P. Then, we have $v_1, v_2, ..., v_p$ which are linearly independent and $Lv_i = 0$ for i from 1 to P. Let $v_l = u$, then we have

$$u^T Lu = \frac{1}{2} \sum_{i,j} w_{ij}(u_i - u_j)^2$$

Since $Lu = 0$, we have

$$\frac{1}{2} \sum_{i,j} w_{ij}(u_i - u_j)^2 = 0$$

As we can see, the summation equals to zero only when all the $w_{ij}(u_i - u_j)^2 = 0$. Thus, if two vertices are connected (i.e. $w_{ij} \neq 0$), then we must have $u_i = u_j$. Thus all the vertices that are connected by a path in a graph should have a constant value in the same vector u. This means that for those with zero values in indicator vector, there is no path ftom this vertex to any other vertex in this "group". Thus, for those vertices with constant values in u, we can say that they form a connected component.
Since one vector u represents a connected components and all such vectors are linearly independent (no overlap on nonzero values), we can conclude that there are P connected components in the graph[2].

4 Spectral Clustering Algorithm

Here we introduce two versions of Spectral Clustering. One is Unnormalized Spectral Clustering and one is Normalized Spectral Clustering

4.1 Unnormalized Spectral Clustering

1. Construct the similarity graph $G = (V, W)$ from the original data points.
2. Determine the K smallest nonzero eigenvalues and corresponding eigenvectors $(v_1, v_2, ..., v_k)$ of L.
3. Construct the matrix $U = [v_1, v_2, ..., v_k]$. U is a $n \times k$ matrix. Treat each row of matrix U as a mapped data point with dimension K and use k-mean algorithm to cluster them into K clusters.

The algorithm achieves the minimum value with respect to *RatioCut*.

Proof. Let a clustered graph V be represented as $V = \{A_1, A_2, ..., A_k\}$. For each A_l for $l \in [1, k]$, construct an indicator vector h_l. The indicator vector is defined as following:

$$h_l = \begin{bmatrix} 0 \\ \vdots \\ \frac{1}{\sqrt{|A_l|}} \\ \frac{1}{\sqrt{|A_l|}} \\ \vdots \\ 0 \end{bmatrix} \tag{6}$$

where $|A_l|$ is the size of cluster A_l and the i-th element in this vector indicate whether the i-th data point is in this cluster(with nonzero value) or not(with zero value). If we replace the x in equation 5 with this indicator vector, we will get

$$h_l^T L h_l = \frac{1}{2} \sum_{i,j} w_{ij} (h_{l_i} - h_{l_j})^2 \tag{7}$$

Because of the way we define the indicator vector, $(h_{l_i} - hl_j)^2$ only have two possible values

- $(h_{l_i} - hl_j)^2 = 0$ when both of them are in cluster A_l or none of them is in the cluster A_l
- $(h_{l_i} - hl_j)^2 = \frac{1}{|A_l|}$ when only one of them is in the cluster A_l

Thus, the equation 7 can be written as following

$$h_l^T L h_l = \frac{1}{2} \sum_{i \in A_l, j \notin A_l or i \notin A_l, j \in A_l} (w_{ij} \frac{1}{|A_l|}) = \frac{Cut(A_l, \bar{A}_l)}{|A_l|} \tag{8}$$

We can easily connect this to the definition of *RatioCut*:

$$RatioCut = \sum_{l=1}^{k} \frac{Cut(A_l, \bar{A}_l)}{|A_l|} = \sum_{l=1}^{k} h_l^T L h_l \tag{9}$$

If we construct a matrix $H = [h_1, h_2, ..., h_k]$, then equation 9 can be written as

$$\sum_{l=1}^{k} h_l^T L h_l = Tr(H^T L H) \tag{10}$$

Since each h_l in the matrix H is the indicator vector for cluster A_l, we have

$$H^T H = I \tag{11}$$

Thus, the problem of minimizing the *RatioCut* becomes the following

$$\min Tr(H^T LH) \text{ subject to } H^T H = I \tag{12}$$

Note: This optimization problem omits one constrain that the columns of H should be indicator vector which means that they should not have overlaps on nonzero values with each other. But, if this constrain is add to this optimization problem, then it will be NP-Complete[2] which is not solvable in realistic time. So, we use a relaxed version here.

According to Rayleigh-Ritz method[4], the resulting H consists of the eigenvectors corresponding to smallest eigenvalues of L.

$$H_{n\times k} = [v_1, v_2, ..., v_k]$$

Since the columns of resulting H will not necessarily be a valid indicator vector, we need an additional operation. We will treat each row of H as the mapped version of original data point in a k-dimensional space and used k-mean algorithm to cluster them into k clusters.

□

Note: If the constrain mentioned in the previous note is added, we can expect that each column of the resulting H will be the same as indicator vector and then, we can cluster them very easily.

4.2 Normalized Spectral Clustering

1. Construct the similarity graph $G = (V, W)$ from the original data points.
2. Determine the K eigenvectors $(v_1, v_2, ..., v_k)$ to $Lv = \lambda Dv$ corresponding to K smallest nonzero eigenvalues.
3. Construct the matrix $U = [v_1, v_2, ..., v_k]$. U is a $n \times k$ matrix. Treat each row of matrix U as a mapped data point with dimension K and use k-mean algorithm to cluster them into K clusters.

Instead of using the eigenvectors of L as in the unnormalized spectral clustering, we use generalized eigenvectors. This algorithm achieve the minimum value with respect to $NormalizedCut$.

Proof. As before, we define an indicator vector h_l for each cluster. The indicator function here is defined

[2]The concept of NP-Completeness is hard to explain in one or two sentence. I think the only thing we need to know here is that if the constraint is added, there is no known way to solve this problem quickly. You can see details about NP-Completeness here[3]

as

$$h_l = \begin{bmatrix} 0 \\ \vdots \\ \frac{1}{\sqrt{|Vol(A_l)|}} \\ \frac{1}{\sqrt{|Vol(A_l)|}} \\ \vdots \\ 0 \end{bmatrix} \tag{13}$$

where $|Vol(A_l)|$ is the sum of d_i. The d_i here is the same as defined in $NomalizedCut$. Following the previous trajectory, we have

$$h_l^T L h_l = \frac{1}{2}\sum_{i,j}(w_{ij}(h_{l_i} - h_{l_j})^2) = \frac{Cut(A_l, \bar{A}_l)}{|Vol(A_l)|} \tag{14}$$

Different from the previous result, we have

$$h_l^T h_l = \frac{|A_l|}{|Vol(A_l)|} \tag{15}$$

If we make some modification by adding the matrix D, we have

$$h_l^T D h_l = 1 \tag{16}$$

Then, the problem will be

$$\min Tr(H^T L H) \ subject\ to\ H^T D H = I \tag{17}$$

To be able to apply Rayleigh-Ritz method again, we let $Q = D^{\frac{1}{2}}H$. Then we have

$$Q^T Q = H^T D^{\frac{1}{2}} D^{\frac{1}{2}} H = I \tag{18}$$

Since $H = D^{-\frac{1}{2}}Q$, we rewrite the optimization problem in equation 17 as

$$\min Tr(Q^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} Q) \ subject\ to\ Q^T Q = I \tag{19}$$

Since we know D and L, this problem becomes the same as that in nonormalized case. So, we need to find the eigenvectors to $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$. We also know that he eigenvectors of $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ satisfy $Lv = \lambda Dv$. Then, the remaining procedures are the same as in nonormalized case.

□

This algorithm has the same problem as the nonormalized case and we still only focus in the relaxed version here.

4.3 Determin K

An simple algorithm to determine the number of clusters is

- Calculate the eigenvalues of corresponding Laplacian matrix and sort them in increasing order
- Find the eigenvalue followed by the biggest gap (gap is the difference between two consecutive eigenvalues)
- Choose the index of this eigenvalue as K

References

[1] Wikipedia contributors. Cluster analysis — Wikipedia, the free encyclopedia, 2019. [Online; accessed 3-November-2019].

[2] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.

[3] Wikipedia contributors. Np-completeness — Wikipedia, the free encyclopedia, 2019. [Online; accessed 7-November-2019].

[4] Wikipedia contributors. Rayleighâritz method — Wikipedia, the free encyclopedia, 2019. [Online; accessed 3-November-2019].