0.1 Spectral Clustering

- *Algorithm:* Unnormalized Spectral Clustering (algo. **??**), Normalized Spectral Clustering (algo. **??**)
- *Input:* Similarity matrix $\mathcal{S} \in \mathbb{R}^{n \times n}$, number k of clusters to construct
- *Complexity:* $\mathcal{O}(n^3)$
- *Data structure compatibility:* N/A
- *Common applications:* machine learning, exploratory data analysis, computer vision and speech processing.

Problem. Spectral Clustering

Precise and concise formal definition of the problem. No long paragraph here, only a few lines.

Description

The similarity of this algorithm to the traditional spectral algorithm and clustering, such as K-means, could be recognized from the name. Spectral determines it will rely on principle components of an input matrix and cluster means it will rely on the data's clustering to make decision. It is originated from graph algorithm. Basically, it considered all the data as points in the space, and all the points could be connected through lines. The point with longer line connected will have lower weight. Then do the cut for the graph, to let the sum of weight in one subgraph as large as possible and let the sum of weight among different subgraphs as small as possible.

So the method used to construct the similarity graph, cut the graph and do the cluster need to be further specified. Some common approaches are listed.

Similarity Graph Construction [**ts**]

There are several popular methods used to transform the data points $x_1, \dots, x_n$ into a graph.

- ϵ-neighborhood graph, usually considered as an unweighted graph. With a threshold ϵ, the distance measured by s_{ij} between x_i and x_j, such that $s_{ij} = ||x_i - x_j||_2^2$. Then each element in W matrix is defined as 0 when $s_{ij} \geq \epsilon$, otherwise defined as ϵ. So many of the information lost in this procedure as only two values are presented.
- k-nearest neighbor graphs. It will traverse all the sample points and use KNN algorithm to get the nearest k points of each data as neighbors. Only these neighbors have $w_{ij} > 0$.
- fully connected graph, because all the weight will be larger than 0. One common used function is Gaussian similarity function, which gives the same similarity graph and weighted graph

$$w_{ij} = s_{ij} = exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2})$$

Cut Method

Min-cut might not give the optimal solution because the weights within one subgraph is not considered. So RatioCut[**slides**] is introduced, which do not consider minimize $cut(A_1, A_2, ...A_k)$, but also maximize the number of points in each subgraph such that

$$RatioCut(A_1, A_2, ...A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{|A_i|}$$

. Laplacian Matrix property is used to solve the equation and the dimension is reduced to finally obtain the result. So a bit information is lost and normally clustering on each line will be used.

Another cut method is Ncut, which will not count the number of points in one subgraph but the $vol(A_i)$

$$NCut(A_1, A_2, ...A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{vol(A_i)}$$

Cluster

The most common final cluster method is $K-$means cluster[**stslides**], which has k centroids that it uses to define clusters. The points are divided into one cluster once the distance it is to this cluster's centroid is smaller than any other centroids.

Algorithm

The most common common spectral clustering algorithm is introduced for unnormalized [**tut**];

```
Algorithm 1: Unnormalized Spectral Clustering
Input  : Similarity matrix $\mathcal{S} \in \mathbb{R}^{n\times n}$, number $k$ of clusters to construct
Output: Cluster $A_1, \dots, A_k$ with $A_i = \{j|y_j \in C_i\}$
1 Construct a similarity graph by one of the ways described above. Let $\mathcal{W}$ be its weighted adjacency matrix
2 Compute the unnormalized Laplacian $\mathcal{L}$
3 Compute the first $k$ eigenvectors $u_1, \dots, u_k$ of $\mathcal{L}$
4 Let $\mathcal{U} \in \mathbb{R}^{n\times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns
5 For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i-$th row of $\mathcal{U}$
6 Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the $k-$means algorithm into clusters $C_1, \dots, C_k$
7 return Cluster $A_1, \dots, A_k$ with $A_i = \{j|y_j \in C_i\}$
```

As the most popular methods used for similarity matrix construction, cut and clustering are fully constructed graph based on the Gaussian function, Ncut and $K-$means, the algorithm is introduced [**tut**]

```
Algorithm 2: Normalized Spectral Clustering
Input  : Similarity matrix $\mathcal{S} \in \mathbb{R}^{n\times n}$, number $k$ of clusters to construct
Output: Cluster $A_1, \dots, A_k$ with $A_i = \{j|y_j \in C_i\}$
1 Construct a similarity graph. Let $\mathcal{W}$ be its weighted adjacency matrix
2 Compute the unnormalized Laplacian $\mathcal{L}$
3 Compute the first $k$ eigenvectors $u_1, \dots, u_k$ of the generalized eigenproblem $\mathcal{L}\sqcap = \lambda Du$
4 Let $\mathcal{U} \in \mathbb{R}^{n\times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns
5 For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i-$th row of $\mathcal{U}$
6 Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the $k-$means algorithm into clusters $C_1, \dots, C_k$
7 return Cluster $A_1, \dots, A_k$ with $A_i = \{j|y_j \in C_i\}$
```

```
Algorithm 3: Name
Input  :
Output:
1 return
```

Time Complexity[**time**]

Most expensive step is computing the eigenvalues/eigenvectors for Laplacian matrix, which is $\mathcal{O}(n^3)$. n is the number of input points. To construct the similarity matrix costs $\mathcal{O}(n^2)$. And the final $K-$means will cost

$\mathcal{O}(nldk)$, where l is the number of $k-$means iteration, d is the dimensionality and k is the final number of clusters.

So the total time complexity is $\mathcal{O}(n^3)$

In conclusion, Spectral Clustering is suitable for sparse data as it only needs the similarity matrix, which is difficult for $K-$means. However, different similarity matrix might give different result.