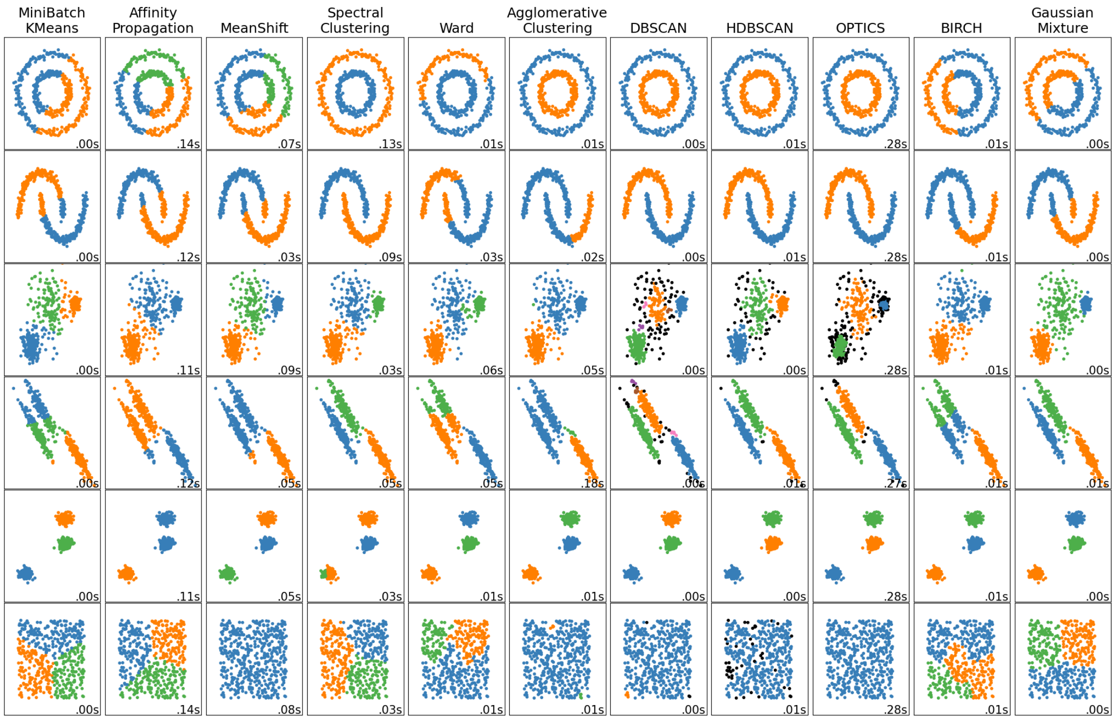
Spectral Clustering Application Documentation

Tony Tan

Aug 2023

Motivation

Spectral Clustering is one of the more superior methods of data clustering for both convex and non convex clusters. It has shown to have higher accuracy than traditional clustering methods like K-means.
Here is an example from scikit-learn[1] to illustrate our argument:



As shown above, Spectral Clustering produces one of the most stable/reliable output throughout all types of data-sets.

1 Application Explanation

This is an educational application designed to demonstrate the concept and application of Spectral Clustering, a powerful technique in the field of machine learning and data science. This technique is used to cluster data into relevant groups. Using our user-friendly interface, you can adjust parameters and observe their impact on the clustering process. For optimal visualization, we employ the make moons data-set from the sci-kit learn library. The adjustable include the datasets, sample size, the number of clusters, and the level of noise in the data, length scale value(σ) for euclidean distance measurement. Users can experiment with various parameters to understand how these changes affect the results of the clustering process.

2 Theory description

Stanford Professor Andrew Ng's paper "On Spectral Clustering"[2] offers deep insights into the mathematical and theoretical foundations of this technique. The paper thoroughly details the algorithmic process, which includes constructing a similarity graph, generating the normalized Laplacian matrix, performing eigen decomposition, and finally, applying k-means clustering on the obtained eigenvectors to produce the clustered dataset. The paper underscores the ability of Spectral Clustering to manage non-convex clusters, a feature that elevates it above traditional methods like K-means for a broad array of applications.

3 Algorithm:

We define a set of points $S = s_1, ..., s_n$ and to partition them into k clusters. The algorithm would construct a series of matrices to represent the connectivity of the graph and we apply eigen-decomposition to the special Laplacian matrix to derive clustering information.
1. Forming the affinity matrix
This is done by forming an affinity matrix A using formula:

$$A_{ij} = exp(-||s_i - s_j||^2/2\sigma^2).$$

The matrix A captures the euclidean distance between each point so we can measure how close each point is to others.

2.1 Forming Diagonal(Degree) Matrix
The diagonal matrix is calculated using:

$$\forall i,\ D_{i,i} = \sum_{j=0}^{n-1} A_{i,j} \quad and \quad \forall j, j \neq i,\ D_{i,j} = 0$$

It is basically a degree matrix to capture how many neighbors each node has and present it along the matrix diagonal.

2.2 Forming Laplacian Matrix
The Laplacian matrix captures the important property of the graph. It retains the information of connectivity between points and also the euclidean distance between points. It's calculated using the following:

$$L = D^{-1/2} A D^{-1/2}$$

Side note:
The above is also called the normalized Laplacian as it normalized the entries when multiplied out. Given that the A adjacency matrix and diagonal matrix are both symmetric matrices, we can also illustrate the Laplacian as below[3]:

$$L_{ij} = \begin{cases} 1, & i = j \\ -\frac{1}{\sqrt{d_i d_j}}, & \text{if } i \text{ is neighbor of } j \text{ and } d_x \text{ is degree of node } x \\ 0 & \text{o.w} \end{cases}$$

3. The k largest eigenvalues and eigenvectors of the Laplacian Matrix
Depend on the number of clusters k we intend to have, we first find all the eigenvalues from the Laplacian matrix $x_1, ..., x_n$. And then we take the k largest eigenvalues from it, namely $y_1, ..., y_k$. Then from $y_1, ..., y_k$, we calculate the corresponding eigenvectors for it, namely $v_1, ..., v_k$. And we form the matrix $Y = [v_1^T, ..., v_k^T]$.

Side note:
The reason we take the k largest eigenvalues and its associate eigenvectors because each of them provides us information about connectivity, partition information of the original graph.

Let's discuss the Fiedler's value(second smallest eigenvalue in unnormalized Laplacian) as an example. From [3], we know that in the unnormalized Laplacian matrix(there is similar equivalence in the normalized Laplacian that we are using), the second smallest eigenvalue, the Fiedler's value is bounded by the Cheeger constant. The lemma is called Cheeger's inequality.

Given graph $G = (V, E)$ and S is a subset of V whose complement is $\overline{S} = V - S$. The Cheeger constant h_G is defined as follow[3]:

$$Vol(S) = \sum_{j \in S} d_j$$

$$CUT(S) = \sum_{j \in S, k \in \overline{S}} A_{jk}$$

$$NCUT(S) = \frac{CUT(S)}{min(Vol(S), Vol(\overline{S}))}$$

$$h_G = min_S NCUT(S)$$

Lemma: Cheeger Inequality[3] states that If G is connected, then

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

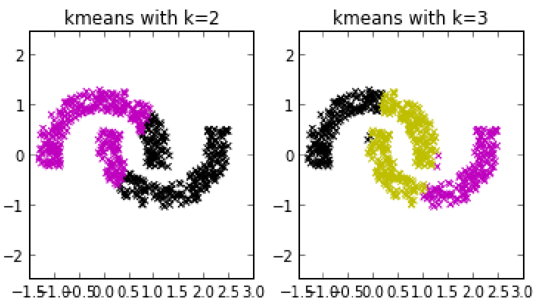
Thus we know that the second smallest eigenvalue is bounded by the minimum normalized cut. Because the N-Cut problem basically asks how to partition the graph using minimum number of cuts. Thus we can derive that the Fiedler's value, varies based on the connectivity of the graph, and thus, it is natural that we can use the Fiedler's vector to derive partition of the graph(particularly into two subsets).[4] There are similar applications to the normalized Laplacian and each eigenvalue to provide information for properly partitioning the graph as well. For example, the largest eigenvalue is bounded by the following lemma[5]:

$$0 \leq min(\frac{CUT(S,\overline{S})}{Vol(S)} + \frac{CUT(S,\overline{S})}{Vol(\overline{S})}) \leq max(\frac{CUT(S,\overline{S})}{Vol(S)} + \frac{CUT(S,\overline{S})}{Vol(\overline{S})}) \leq \lambda_n(L)$$

From the above, we can derive there is relationship between the largest eigenvalue in the normalized Laplacian with the NCut problem as well. So we know there's similar nature of the Fiedler's value of the unnormalized Laplacian compares to the largest eigenvalues of the normalized Laplacian. It also shows that the eigenstructure of the Laplacian reveals properties about the graph's structure which is why we choose to use the k largest eigenvalues to partition our graph.

4. Normalize eigenvectors and apply K-means[6]
We then form the normalized matrix N by performing normalization on the matrix Y above. We do this by making each row from Y to have unit length(i.e. normalize Y to become N). Namely, $N_{i,j} = Y_{i,j}/(\sum_j Y_{i,j}^2)^{1/2}$. We treat each row from N as a point, and we apply K-means to N to obtain the labels by using K-means. Each row represent one point, associated with its characteristics in the eigenspace, as the eigenvectors' entries represent a point's information in its eigenspace. Given that K-means is a centroid based clustering method, so the closer the points are after eigen-decomposition(i.e. their eigen-spaces representation), meaning they are more likely to have the same characteristics, similarity, closeness in the original graph.

Tho a reasonable question would be if we are applying K-means here, why don't we applying k-means directly at the beginning? It turns out that K-means is good at partitioning data with convex clusters while not optimal with the non-convex counter-parts. As K-means is a centroid-based partition method, where it finds a centroid and try to assign points that is closest to the centroid to partition the graph. But for non-convex data, the below is an example to illustrate how K-means performs in clustering:



Apparently, the above is not the optimal cluster. Therefore, we use the spectral method which perform well in non-convex data-sets as well.

5. Cluster assignment
Lastly, we use the labels' information from applying K-means above to assign each original point a cluster. Namely, we assign point s_i to cluster j if and only if K-means assigned row i of the matrix N to cluster j.

4 Application

4.1 Stack

The application is made using html, css, javascript with Flask backend.

4.2 Frontend

The front-end consists of homepage, step1 to step5 explanation and all-in-one explanation. The application also leverages the the extensive list of pre-designed components from Bootstrap version 4.5.2.

4.3 Back-end

The backend is done using Flask to handle $Post/GET$ requests with libraries from sci-kit learn data-sets, numpy/scipy for matrix calculation and matplotlib for graph display.

5 Extra resources for readings:

5.1 Textbooks

To perfect the theory-explanations in the application. I believe there are more background readings need to be completed in order to explain the algorithm in better/more intuitive way. One of the most important textbook which makes normalized Laplacian spectral clustering famous is: Fan R. K. Chung Graph Theory[7].

5.2 Papers

The research from Andrew. Ng's "On spectral clustering"[2] and Normalized cuts and image segmentation[4] are a must read for people who are interested in spectral clustering. The PHD thesis from professor. Michael Scott Cavers provides an in depth analysis of the normalized Laplacian matrix as well[5].

5.3 Others

A insightful Quora answer: by Muni Sreenivas Pydi, etc.

6 Conclusion:

We went over why spectral clustering is one of the more superior data clustering method becuase of its ability to deal with non-convex clusters. We also went over a step-by-step analysis of the algorithm from building affinity matrix, normalized Laplacian matrix, extracting k-largest eigenvalues with its associated eigenvectors, and to applying K-means at the end to extract the labelling information. We also went over some technology we used the build the application and introduced some extra resources for readings if users are interested.
Words from the author:
The spectral clustering is a very big topic with many mathematical derivations behind(VERY DEEP). This guide only serves as a tip of the iceberg to demonstrate one of its application in the real life scenario and tried its best to give a general overview of some insights behind the algorithm. Hope it helped :)

7 References:

[1] sklearn. (n.d.). Scikit-learn. scikit. `https://scikit-learn.org/stable/`
[2] Ng, A. Y., Jordan, M. I., &; Weiss, Y. (n.d.). On spectral clustering: Analysis and an algorithm - neurips. `https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf`
[3] Y. Yao, "Spectral Graph Theory," Department of Mathematics, Peking University, 2011. `https://www.math.pku.edu.cn/teachers/yaoy/Fall2011/lecture08.pdf`
[4]Shi, J., &; Malik, J. (n.d.). Normalized cuts and image segmentation — IEEE Journals & Magazine `https://ieeexplore.ieee.org/abstract/document/868688/`
[5]Cavers, M. S. (n.d.). The normalized Laplacian matrix and general Randić index of graphs. `https://librarysearch.library.utoronto.ca/discovery/fulldisplay?context=PC&vid=01UTORONTO_INST:UTORONTO&search_scope=UTL_AND_CI&tab=Everything&docid=cdi_proquest_journals_1069374864`
[6] Wikipedia Contributors. (2023, August 7). K-means clustering. Wikipedia. `https://en.wikipedia.org/wiki/K-means_clustering`
[7] Chung, F. R. K. (n.d.). Lectures on spectral graph theory fan R. K. Chung. `https://mathweb.ucsd.edu/~fan/research/cbms.pdf`