How does Spectral Clustering work?

In spectral clustering, the data points are treated as nodes of a graph. Thus, clustering is treated as a graph partitioning problem. The nodes are then mapped to a low-dimensional space that can be easily segregated to form clusters. **An important point to note is that no assumption is made about the shape/form of the clusters.**

What are the steps for Spectral Clustering?

Spectral clustering involves 3 steps:

1. Compute a similarity graph

2. Project the data onto a low-dimensional space

3. Create clusters

Step 1 — Compute a similarity graph:

We first create an undirected graph G = (V, E) with vertex set V = {*v1, v2, …, vn*} = 1, 2, …, n observations in the data. This can be represented by an adjacency matrix which has the similarity between each vertex as its elements . To do this, we can either compute:

1) The ε-neighborhood graph: Here we connect all points whose pairwise distances are smaller than ε. As the distances between all connected points are roughly of the same scale (at most ε), weighting the edges would not incorporate more information about the data to the graph. Hence, the ε-neighborhood graph is usually considered as an unweighted graph.

2) KNN Graph: Here we use K Nearest Neighbors to connect vertex *vi* with vertex *vj* if *vj* is among the k-nearest neighbors of *vi*.

But we may have an issue if the nearest neighbors are not symmetric, i.e. if there is a vertex *vi* which has *vj* as a nearest neighbor, it is not necessary that *vi* is a nearest neighbor of *vj*. Thus, we end up getting a directed graph which is a problem as we do not know what similarity between 2 points means in that case. There are two ways of making this graph undirected.

The first way is to simply ignore the directions of the edges, i.e. we connect *vi* and *vj* with an undirected edge if *vi* is among the k-nearest neighbors of *vj* or if *vj* is among the k-nearest neighbors of *vi* . The resulting graph is what is usually called the k-nearest neighbor graph.

The second choice is to connect vertices *vi* and *vj* if both *vi* is among the k-nearest neighbors of *vj* and *vj* is among the k-nearest neighbors of *vi* . The resulting graph is called the mutual k-nearest neighbor graph.

In both cases, after connecting the appropriate vertices we weight the edges by the similarity of the adjacent points.

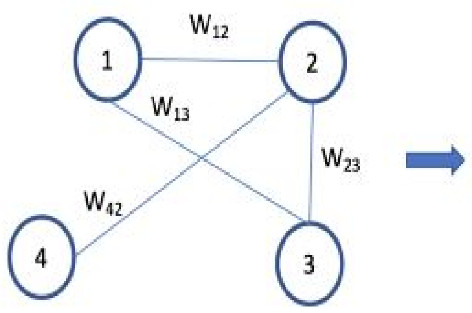
3) Fully connected graph: To construct this graph, we simply connect all points with each other, and we weight all edges by similarity *sij*. This graph should model the local neighborhood relationships, thus similarity functions such as Gaussian similarity function are used.

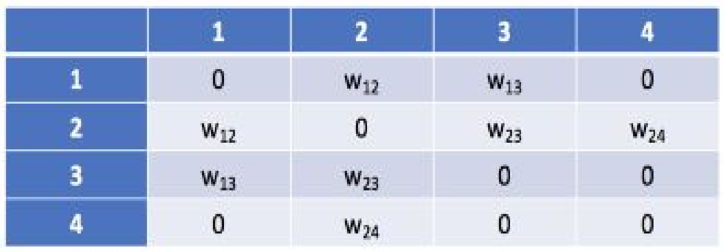
$$s(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$$

Here the parameter σ controls the width of the neighborhoods, similar to the parameter ε in case of the ε-neighborhood graph.

Thus, when we create an adjacency matrix for any of these graphs, $Aij \sim 1$ when the points are close and $Aij \to 0$ if the points are far apart.

Consider the following graph with nodes 1 to 4, weights (or similarity) wij and its adjacency matrix:





	1	2	3	4
1	0	W_{12}	W_{13}	0
2	W_{12}	0	W_{23}	W_{24}
3	W_{13}	W_{23}	0	0
4	0	W_{24}	0	0

Step 2 — Project the data onto a low-dimensional space:

Data points in the same cluster may also be far away–even farther away than points in different clusters. Our goal then is to transform the space so that when the 2 points are close, they are always in same cluster, and when they are far apart, they are in different clusters. We need to project our observations into a low-dimensional space. For this, we compute the Graph Laplacian, which is just another matrix representation of a graph and can be useful in finding interesting properties of a graph. This can be computed as:

$$L = D - A,$$
where A is the adjacency matrix and D is the degree matrix such that

$$d_i = \sum_{\{j|(i,j)\epsilon E\}} w_{ij} \qquad Thus, L_{ij} = \begin{cases} d_i & if\ i = j \\ -w_{ij} & if\ (i,j)\ \epsilon\ E \\ 0 & if\ (i,j) \notin E \end{cases}$$

$$\begin{aligned} d_1 &= w_{12} + w_{13} \\ d_2 &= w_{12} + w_{23} + w_{24} \\ d_3 &= w_{12} + w_{23} \\ d_4 &= w_{24} \end{aligned} \quad \Rightarrow \quad L = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & d_1 & -W_{12} & -W_{13} & 0 \\ 2 & -W_{12} & d_2 & -W_{23} & -W_{24} \\ 3 & -W_{13} & -W_{23} & d_3 & 0 \\ 4 & 0 & -W_{24} & 0 & d_4 \end{array}$$

The whole purpose of computing the Graph Laplacian L was to find eigenvalues and eigenvectors for it, in order to embed the data points into a low-dimensional space. So now, we can go ahead and find eigenvalues. We know that:

$$\boldsymbol{L\lambda = \lambda v}$$

where v is the eigenvector of L corresponding to eigenvalue λ.
Thus we get eigenvalues $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ where $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ and eigenvectors $\{v_1, v_2, \dots, v_n\}$.

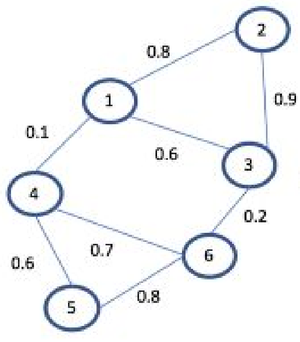
1. If L has eigenvalue 0 with k different eigenvectors, such that $0 = \lambda_1 = \lambda_2 = \dots = \lambda_k$, graph G has k connected components.

2. If G is connected, i.e. $0=\lambda_1$ and $\lambda_2>0$, λ_2 is the algebraic connectivity of G. Thus, greater λ_2, greater the connectivity.

Spectrum of the Laplacian

$$\boldsymbol{0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n}$$

$G =$



$\Rightarrow$ $L =$

	1	2	3	4	5	6
1	1.5	-0.8	-0.6	-0.1	0	0
2	-0.8	1.7	-0.9	0	0	0
3	-0.6	-0.9	1.7	0	0	-0.2
4	-0.1	0	0	1.4	-0.6	-0.7
5	0	0	0	-0.6	1.4	-0.8
6	0	0	-0.2	-0.7	-0.8	1.7

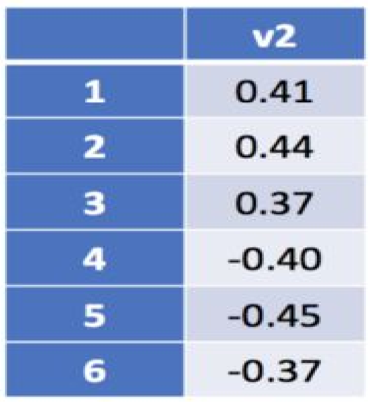
We then compute eigenvalues and eigenvectors for L.

Step 3 — Create clusters:

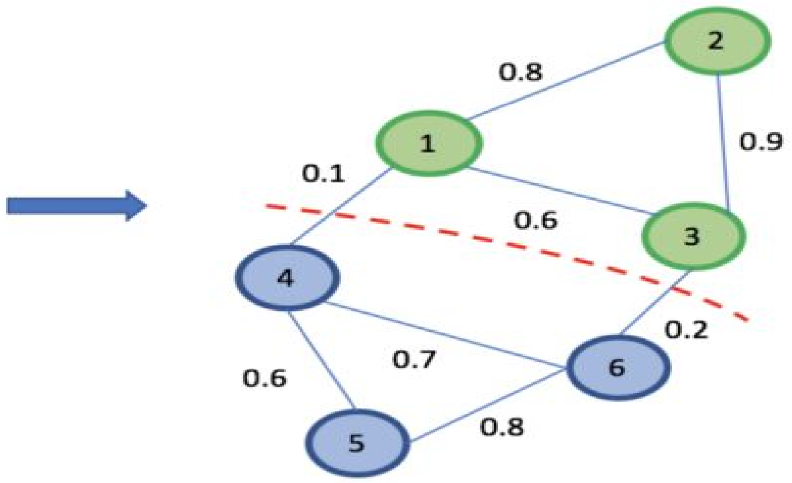
For this step, we use the eigenvector corresponding to the 2nd eigenvalue to assign values to each node. On calculating, the 2nd eigenvalue is 0.189 and the corresponding eigenvector v2 = [0.41, 0.44, 0.37, -0.4, -0.45, -0.37].

To get bipartite clustering (2 distinct clusters), we first assign each element of v2 to the nodes such that *{node1:0.41 , node2:0.44 , … node6: -0.37}*. We then split the nodes such that all nodes with value > 0 are in one cluster, and all other nodes are in the other cluster. Thus, in this case, we get nodes 1, 2 & 3 in one cluster, and 4, 5 & 6 in the 2nd cluster.

It is important to note that the 2nd eigenvalue indicates how tightly connected the nodes are in the graph. For good, clean partitioning, lower the 2nd eigenvalue, better the clusters.



	v2
1	0.41
2	0.44
3	0.37
4	-0.40
5	-0.45
6	-0.37



For k clusters, we have to modify our Laplacian to normalize it.

Thus we get:

1. For k clusters, compute the first k eigenvectors $\{v_1, v_2, \dots, v_k\}$.
2. Stack the vectors vertically to form a matrix with the vectors as columns.
3. Represent every node by the corresponding row of this new matrix. These rows form the feature vectors of the nodes.
4. Use K-Means Clustering to now cluster these points into k clusters $\{C_1, C_2, \dots, C_k\}$.

Advantages and Disadvantages of Spectral Clustering

Advantages:

1. Does not make strong assumptions on the statistics of the clusters — Clustering techniques like K-Means Clustering assume that the points assigned to a cluster are spherical about the cluster center. This is a strong assumption to make, and may not always be relevant. In such cases, spectral clustering helps create more accurate clusters.

2. Easy to implement and gives good clustering results. It can correctly cluster observations that actually belong to the same cluster but are farther off than observations in other clusters due to dimension reduction.

3. Reasonably fast for sparse data sets of several thousand elements.

Disadvantages:

1. Use of K-Means clustering in the final step implies that the clusters are not always the same. They may vary depending on the choice of initial centroids.

2. Computationally expensive for large datasets — This is because eigenvalues and eigenvectors need to be computed and then we have to do clustering on these vectors. For large, dense datasets, this may increase time complexity quite a bit.

References

1. https://www.youtube.com/watch?v=zkgm0i77jQ8
2. https://youtu.be/uxsDKhZHDcc
3. https://www.geeksforgeeks.org/ml-spectral-clustering/
4. https://ocw.mit.edu/courses/mathematics/18-409-topics-in-theoretical-computer-science-an-algorithmists-toolkit-fall-2009/lecture-notes/MIT18_409F09_scribe1.pdf
5. https://ocw.mit.edu/courses/mathematics/18-409-topics-in-theoretical-computer-science-an-algorithmists-toolkit-fall-2009/lecture-notes/MIT18_409F09_scribe2.pdf
6. https://ocw.mit.edu/courses/mathematics/18-409-topics-in-theoretical-computer-science-an-algorithmists-toolkit-fall-2009/lecture-notes/MIT18_409F09_scribe3.pdf
7. https://juanitorduz.github.io/spectral_clustering/

GMM

1. http://www.cse.iitm.ac.in/~vplab/courses/DVP/PDF/gmm.pdf
2. https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/
3. https://www.youtube.com/watch?v=DODphRRL79c
4. https://www.geeksforgeeks.org/gaussian-mixture-model/