Machine Learning for Data Science (CS 4786)

Lecture 13-14: Spectral Clustering

The text in black outlines high level ideas. The text in blue provides simple mathematical details to "derive" or get to the algorithm or method. The text in red are mathematical details for those who are interested.

1 Similarity based clustering

What if we are only provided similarities or distances between the points we want to cluster. The data points need not even live in a vector space.
Specifically for the n points we are given an $n \times n$ matrix A of similarities where $A_{i,j} = A_{j,i} \geq 0$ indicates how similar points i and j are.

Example 1: If distances $d(x_i, x_j)$ are provided to us, one can also use adjacency matrix of a "nearest neighbor graph".

- For each element $i \in [n]$ pick the k-nearest neighbors of data point i amongst the n points according to distances provided.
- Form a graph G with n vertices by inducing an edge between every point and its k-nearest neighbor.
- Use as similarity matrix A, the adjacency matrix of this graph.

Example 2: If distances $d(x_i, x_j)$ are provided to us, one can use

$$A_{i,j} = \exp\left(-d(x_i, x_j)^2\right)$$

as an indication of similarity.
Example 3: Friendship/acquaintance graph from social network sites.

Can we still cluster the points?

Spectral Clustering Overview:

1. Input: Matrix A of size $n \times n$ indicating similarity
2. Embed the n points into low, K dimensional space to get "data" matrix X with n points, each in K dimensions.
3. Perform k-means algorithm on these n points.

2 Graph Clustering and Laplacian Matrix

Simplest example of a similarity matrix on can consider is the adjacency matrix of an unweighted undirected graph.

$$A_{i,j} = \begin{cases} 1 & \text{if edge } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The **Laplacian matrix** for the graph is defined as follows:

$$L = D - A$$

where D is a diagonal matrix with $D_{i,i} = \sum_{j=1}^{n} A_{i,j} = \text{degree}(i)$.

Ideal Example:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \quad L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & -2 \end{bmatrix}$$

Eigenvector with smallest Eigenvalue, picks out the cluster!

$$\mathbf{w}_1 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ 0.5774 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

For a connected graph, exactly one, the smallest one of the eigenvalues of L is 0. Corresponding eigenvector is $\mathbf{1}/\sqrt{n} = (1/\sqrt{n}, \ldots, 1/\sqrt{n})^\top$.

Why is this the case? Proof.

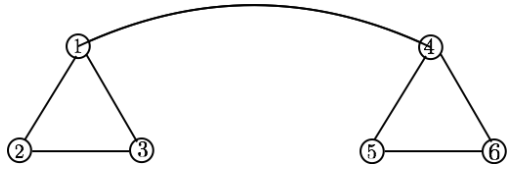
For any graph, the number of eigenvalues that have value of 0 is equal to the number of connected components. Corresponding eigenvector are (normalized) incidence vectors indicating the connected components.

Why is this? Proof.

For the ideal case, the eigenvectors corresponding to smallest eigenvalues give us a nice embedding using which we can cluster (perfectly).

3 Cuts and Graph Clustering

What happens when we add an edge to our ideal example?



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad , \quad L = \begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & -2 \end{bmatrix}$$

Does the eigenvector corresponding to the smallest eigenvalue help?

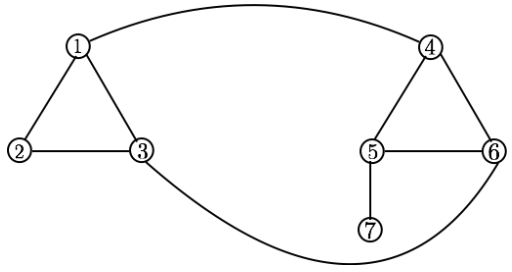
Intuitively we want to pick the partition that cuts the smallest number of edges.
Why does this involve the Laplacian matrix?

Say for now we are interested in two clusters and say $c_i \in \{0, 1\}$ indicates the cluster assignment given to point i. That is $c_i = 0$ means we put point i in cluster 0 and $c_i = 1$ means we put it in cluster 1. Now $\sum_{(i,j)\in E}(c_i - c_j)^2$ indicates number of edges cut. Note that:

$$\frac{1}{2}\sum_{(i,j)\in E}(c_i - c_j)^2 = \frac{1}{2}\sum_{(i,j)\in E}\left(c_i^2 + c_j^2 - 2c_ic_j\right) = \left(\sum_{i\in V} D_{i,i}c_i^2 - c_ic_jA_{i,j}\right) = c^\top Dc - c^\top Ac = c^\top Lc$$

Minimizing $c^\top Lc$ gives us assignment with small cut value. Roughly speaking, eigenvectors corresponding to smallest values are indicators of cluster assignment.

Is there a problem? Consider the following example where we have added one other vertex and two edges more to our graph.



How should we cluster this graph? Should vertex 7 be one cluster and remaining another cluster? Or do we group $1, 2$ and 3 in one cluster and the remaining in another cluster? What is this was a massive graph and there was this poor loner vertex 7 hanging out?

3.1 Normalized Cut

Intuition: Find partitions that minimize number of cuts while ensuring that number of edges within each partition is large.

Roughly speaking:

$$\begin{aligned} &\text{Minimize} \quad c^\top L c \\ &\text{s.t.} \quad c^\top D c = 1 \end{aligned}$$

Why does this objective correspond to our intuition? What would the embedding looks like?

The normalized Laplacian is defined as:

$$\tilde{L} = I_{n \times n} - D^{-1/2} A D^{-1/2}$$

Embedding: take the eigenvectors corresponding to the K smallest eigenvectors of the matrix $\tilde{L}$.