EECS 445 Discussion 9: Spectral Clustering, Hierarchical Clustering, and Collaborative Filtering

March 24, 2021

1 Spectral Clustering

Spectral clustering is, at its core, a graph partitioning algorithm. Specifically, we consider each data point to be a node and construct edges connecting these nodes. We define $S_{i,j}$ to be the (symmetric) similarity between points $\bar{x}^{(i)}$ and $\bar{x}^{(j)}$. One common technique for determining similarity is the cosine similarity metric defined in equation (1).

$$S_{i,j} = \frac{\bar{x}^{(i)} \cdot \bar{x}^{(j)}}{\|\bar{x}^{(i)}\|_2 \|\bar{x}^{(j)}\|_2} \tag{1}$$

We weight the edges of the graph with their similarity value, and prune edges for which the similarity is below some thresholded value ϵ to obtain the graph adjacency matrix.

$$w_{i,j} = \begin{cases} S_{i,j}, & \text{if } S_{i,j} > \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The problem of clustering is then equivalent to finding a partitioning of the graph into k connected components that minimizes the cost of those cuts. Define the diagonal matrix D such that $D_{i,i}$ is the degree of graph node i and $D_{i,j} = 0$ for $j \neq i$.

$$D_{i,i} = \sum_j w_{i,j} \tag{3}$$

We construct the unnormalized graph Laplacian L as follows.

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

Intuitively, we want to split our graph in a manner that has a minimal effect on the values of the graph Laplacian. Thus, we wish to perform our partition on the k eigenvectors corresponding with the smallest k eigenvalues of L. Because L is symmetric and positive semi-definite, we know that these eigenvalues must be non-negative and real. Let $\bar{u}^{(i)} \in \mathbb{R}^n$ be the eigenvector corresponding to the ith smallest eigenvalue and define the matrix U as follows.

$$U = \begin{bmatrix} | & | & & | \\ \bar{u}^{(1)} & \bar{u}^{(2)} & \dots & \bar{u}^{(k)} \\ | & | & & | \end{bmatrix} \tag{5}$$

We can then find a clustering of our original points by applying another clustering algorithm (e.g., k-means) to the *row* vectors of U. This clustering represents both our optimal graph cut approximation as well as the clustering of data points within the input space.

1.1 Spectral Clustering as a Graph Cut Optimization Problem

Now let's look at some of the graph partitioning mathematics behind spectral clustering. For a weighted graph G, a **cut** is defined as a partitioning of the vertices of G into two sets, A and $\overline{A}$. The **cost of the cut** is the sum of the weights of the edges whose endpoints fall in the two sets:

$$cut(A, \overline{A}) = \sum_{u \in A, v \in \overline{A}} w_{uv}$$

where w_{ij} is the weight of the edge between vertices i and j. Note that the definition of a partioning implies that $A \cap \overline{A} = \emptyset$ and $A \cup \overline{A} = V$, where V is the vertex set of G, i.e. with a single cut we are splitting the vertices into two groups.

We extend the definition of the cost of a cut to the cost of a partitioning into k sets by defining the **ratio cut**:

$$cut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{|A_i|}$$

In spectral clustering our weighted graph lets each datapoint be a vertex and each edge be weighted by a similarity score between its endpoints. Thus we can approximate the clusters by looking for graph partitions that minimize the ratio cut of paritioning.
For example, let $k = 2$. We can define our partitioning as some vector $f \in \{-1, 1\}^n$, where a value of 1 indicates the point is in one cluster while a value of -1 indicates the point is in the other cluster. Now we can compute the cost of the partitioning defined by f as followed:

$$R = \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2$$

For simplicity we have omitted the normalization by the size of each set in the cost (see the original definition of ratio cut) from this formulation. Now, if we define W to be our adjacency (weights) matrix, D to be the degree matrix such that D is zero off of the diagonal and ith element along the diagonal is the degree of the ith vertex, then $L = D - W$ is called the Laplacian matrix. As it turns out, $R = f^T L f$ so we can reformulate our problem as finding

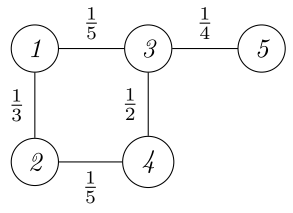
$$f^* = \underset{f \in \{-1,1\}^n}{\arg\min} \ f^T L f$$

This is NP-hard, so spectral clustering relaxes the formulation by allowing f to take on any real value:

$$f^* = \underset{f \in \mathbb{R}^n}{\arg\min} \, f^T L f$$

and performs the minimization by clustering the rows of the matrix formed by the two eigenvectors of L that have the lowest eigenvalues. In general, to find k clusters, we cluster the rows of the matrix formed by the k eigenvectors of L with the lowest eigenvalues as shown in section 1.

Discussion Question 1. *Apply spectral clustering to the following graph. Find the clusters corresponding to $k = 2$ and $k = 3$.*



Solution: We first construct our similarity matrix W and the corresponding degree matrix D:

$$W = \begin{bmatrix} 1 & \frac{1}{3} & \frac{1}{5} & 0 & 0 \\ \frac{1}{3} & 1 & 0 & \frac{1}{5} & 0 \\ \frac{1}{5} & 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{5} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} \frac{23}{15} & 0 & 0 & 0 & 0 \\ 0 & \frac{23}{15} & 0 & 0 & 0 \\ 0 & 0 & \frac{39}{20} & 0 & 0 \\ 0 & 0 & 0 & \frac{17}{10} & 0 \\ 0 & 0 & 0 & 0 & \frac{5}{4} \end{bmatrix}$$

From these matrices, we compute our graph Laplacian:

$$L = \begin{bmatrix} \frac{8}{15} & -\frac{1}{3} & -\frac{1}{5} & 0 & 0 \\ -\frac{1}{3} & \frac{8}{15} & 0 & -\frac{1}{5} & 0 \\ -\frac{1}{5} & 0 & \frac{19}{20} & -\frac{1}{2} & -\frac{1}{4} \\ 0 & -\frac{1}{5} & -\frac{1}{2} & \frac{7}{10} & 0 \\ 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix}$$

Using the `np.linalg.eigh` method to solve for the eigenvectors, we find that:

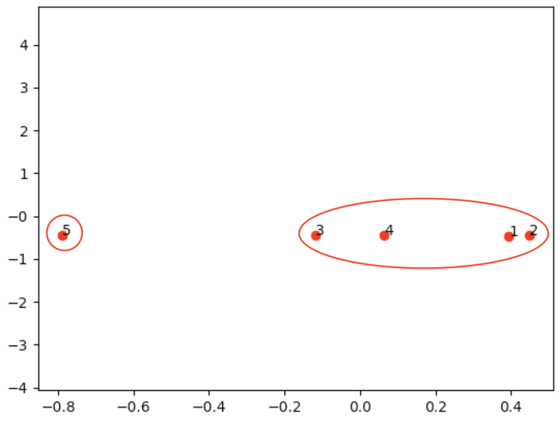
$$U = \begin{bmatrix} -0.4472136 & 0.39424509 & 0.4344079 & 0.63026607 & -0.24212655 \\ -0.4472136 & 0.44976909 & 0.26010872 & -0.69647415 & 0.21207303 \\ -0.4472136 & -0.11769636 & -0.3720193 & 0.29966451 & 0.74696077 \\ -0.4472136 & 0.06384072 & -0.68719191 & -0.0988583 & -0.56028446 \\ -0.4472136 & -0.79015853 & 0.36469459 & -0.13459812 & -0.15662279 \end{bmatrix}$$

These eigenvectors are sorted by increasing eigenvalue.

The eigenvalues are shown below.

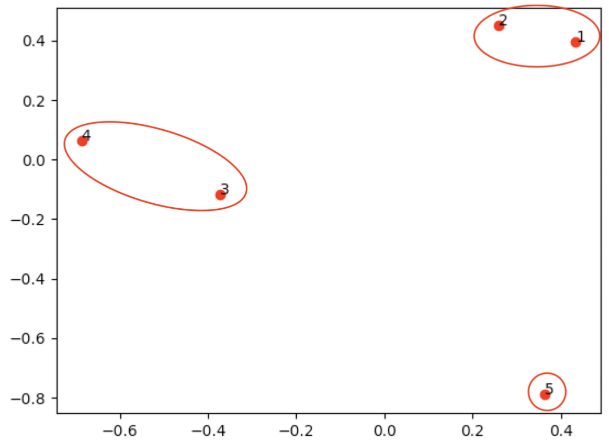
$$\begin{bmatrix} 0 & 0.212761788 & 0.505021126 & 0.806591182 & 1.44229257 \end{bmatrix}$$

For $k = 2$, we look at the first 2 columns of U. We can plot each row as a point in $\mathbb{R}^2$:



Now we can determine the clusters by running K-means on the data points above with $K = 2$. We get points 1, 2, 3, and 4 in one cluster, and point 5 in the other cluster.

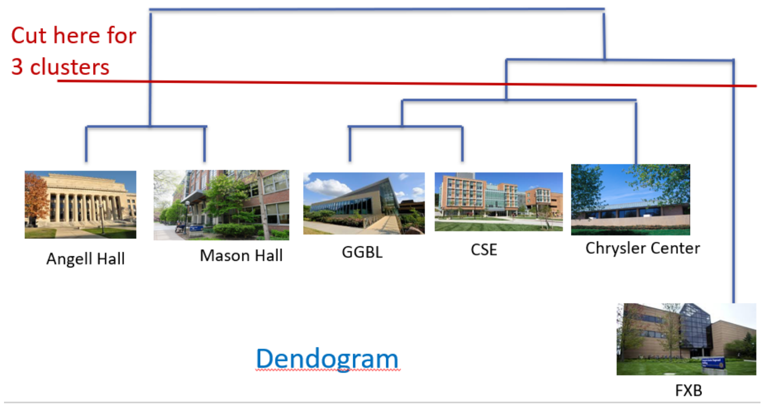
For $k = 3$, we look at the first 3 columns of U. Since the first coordinate is the same for each point, we can ignore it. Thus, we can plot the remaining two columns in $\mathbb{R}^2$ in the same way:



Similar to before, we can determine the clusters by running K-means on the data points above with $k = 3$. We get points 1 and 2 in one cluster, points 3 and 4 in another cluster, and point 5 in the third cluster.

2 Hierarchical Clustering

In hierarchical clustering, we build a hierarchy of the points by either starting with every point in its own cluster and then repeatedly merging clusters together based on some criteria ("agglomerative clustering") or by starting with one big cluster and then repeatedly splitting clusters up based on some criteria ("divisive clustering"). The hierarchy formed by this process can be visualized as a dendogram:



We can then use the information in the hierarchy to perform our clustering, as shown in the figure. The following are some common linkage criteria that are used for determining how to split or merge clusters, where $d(\bar{x}, \bar{y})$ is some distance measure between two vectors $\bar{x}$ and $\bar{y}$:

Single-linkage	$D(C_i, C_j) = \min_{\bar{x} \in C_i, \bar{y} \in C_j} d(\bar{x}, \bar{y})$
Complete-linkage	$D(C_i, C_j) = \max_{\bar{x} \in C_i, \bar{y} \in C_j} d(\bar{x}, \bar{y})$
Average-linkage	$D(C_i, C_j) = \frac{1}{\lvert C_i \rvert} \frac{1}{\lvert C_j \rvert} \sum_{\bar{x} \in C_i, \bar{y} \in C_j} d(\bar{x}, \bar{y})$

```
// Assign each point to its own cluster:
Initialization:  Set $C_i = \{\bar{x}^{(i)}\}$ for $i = 1 \dots n$
    while convergence criteria not met:
        $i, j = \arg\min_{i,j} D(C_i, C_j)$
        Merge $C_i$ and $C_j$
    end while
```

You can choose a variety of convergence criteria: For example could finish when there are only k clusters left or when the similarity between clusters is smaller than some threshold.

Discussion Question 2. *Apply agglomerative clustering to the following data set using complete-linkage and draw the resulting dendogram. The distance matrix for the data set has been provided below. Assume the convergence criteria is when there is 1 cluster remaining.*

Clusters	*1*	*2*	*3*	*4*	*5*	*6*
1	*0*	*3*	*7*	*9*	*5*	*6*
2	*3*	*0*	*2*	*4*	*8*	*14*
3	*7*	*2*	*0*	*11*	*1*	*10*
4	*9*	*4*	*11*	*0*	*17*	*15*
5	*5*	*8*	*1*	*17*	*0*	*12*
6	*6*	*14*	*10*	*15*	*12*	*0*

Solution: Following the pseudocode for the agglomerative algorithm above, at each iteration, we need to choose the two clusters which have the minimum distance between them, then combine them into a single cluster. The distance from this new cluster to all of the other clusters will then have to be recomputed using the chosen linkage scheme. Since we are using complete-linkage, we will compute the distance between clusters by finding the maximum distance between all of the corresponding data points in each cluster.

Iteration 1: $i, j = 3, 5$. The distance matrix for the new cluster assignments is:

Clusters	1	2	4	6	3,5
1	0	3	9	6	7
2	3	0	4	14	8
4	9	4	0	15	17
6	6	14	15	0	12
3,5	7	8	17	12	0

Iteration 2: $i, j = 1, 2$. The distance matrix for the new cluster assignments is:

Clusters	1,2	4	6	3,5
1,2	0	9	14	8
4	9	0	15	17
6	14	15	0	12
3,5	8	17	12	0

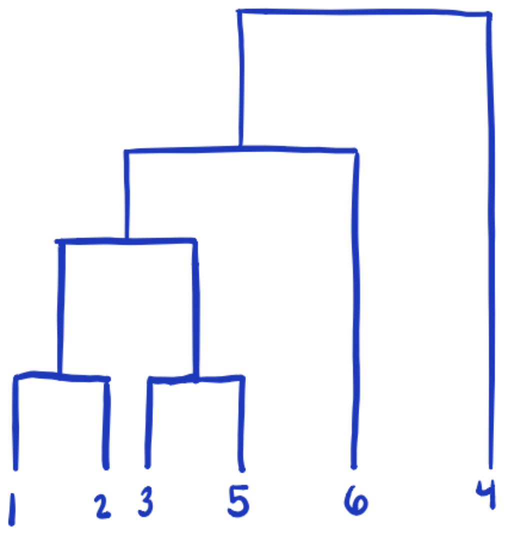
Iteration 3: $i, j = (1, 2), (3, 5)$. The distance matrix for the new cluster assignments is:

Clusters	(1,2),(3,5)	4	6
(1,2),(3,5)	0	17	14
4	17	0	15
6	14	15	0

Iteration 4: $i, j = ((1,2),(3,5)), 6$. The distance matrix for the new cluster assignments is:

Clusters	((1,2),(3,5)), 6	4
((1,2),(3,5)), 6	0	17
4	17	0

Thus, if we combine these last two clusters together, we get the following dendogram for this dataset:



3 Collaborative Filtering

Recommendation systems were historically the result of substantial human effort. For example, the music service Pandora used to hire experts to create curated playlists that corresponded to specific genres and moods. Collaborative filtering both negates the need for substantial expert knowledge and improves upon the quality of recommendations by solving a regression problem for the expected rating that a user would give to a piece of content using only a metric of similarity between users. For instance, if user A and user B liked very similar music and user A likes song X that B has not heard, a collaborative filtering algorithm would conclude that user B may also like song X. This is the same technique currently used by corporations such as Spotify and Netflix.

Here we will focus on the matrix factorization technique for collaborative filtering. Consider the case of n users and m possible song selections. Let Y_{ai} be the observed rating that user a assigned to song i, where $Y_{ai} \in \{-1, 0, 1\}$. We wish to construct an approximation $\hat{Y}_{ai}$ for *all* values of a, i to satisfy the following optimization problem.

$$\min_{\hat{Y}} J(\hat{Y}) = \min_{\hat{Y}} \frac{1}{2} \sum_{(a,i)\in D} (Y_{ai} - \hat{Y}_{ai})^2 + \frac{\lambda}{2} \sum_{a,i} (\hat{Y}_{ai})^2 \tag{6}$$

Here, D represents the set of all observed ratings and λ is a tuneable hyperparameter. Note that without any constraint on $\hat{Y}$, we end up with the trivial solution that $\hat{Y}_{ai} = 0$ for all unobserved values.