10 Spectral Clustering

Another perspective on clustering is that there are three main types: (1) *Bottom-Up*, (2) *Assignment-Based*, and (3) *Top-Down*. The bottom-up variety was like the hierarchical agglomerative clustering where we start with very small clusters and build bigger clusters. The assignment based clustering was like the k-center or the k-means variety were we "assign" each object to a center. Given the centers, there is no need to build or carve the clusters. The third type, top-down clustering, is what we will be discussing here. It starts from one big cluster and gradually divides the big clusters into smaller and smaller clusters.

At a high level the idea of top down clustering can be described very easily.

- Find the best cut of the data into two pieces.
- Recur on both pieces until that data should not be split anymore.

What remains is to determine the best way to split a set into two pieces. Then finding a threshold has similar options as with Hierarchical clustering.

Also we will need to discuss graphs, and perform clustering on graphs.

10.1 Graphs

A graph is an *abstract data type* that may seem very natural once you are familiar with and used to it. But if it is new, it may take a while to sink in. We will revisit them many times in the class.

A graph $G = (V, E)$ is defined by a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$ where each edge e_j is an unordered (or ordered in a directed graph) pair of edges: $e_j = \{v_i, v_{i'}\}$.

Two vertices v_1 and v_k are *connected* if there is a sequence of edges $\langle e_1, \ldots, e_{k-1}\rangle$ such that e_1 contains v_1, e_{k-1} contains v_k, and each consecutive edges can be ordered so $e_j = \{v_i, v_{i+1}\}$ and $e_{j+1} = \{v_{i+1}, v_{i+2}\}$ where the second element in e_j is the same as the first in e_{j+1}.

Consider an example graph portrayed three ways.

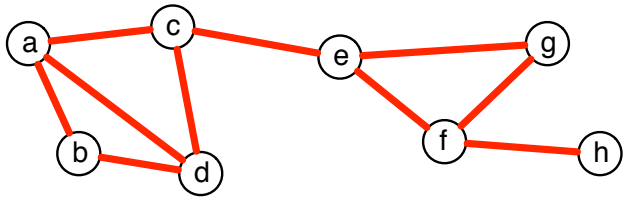
Mathematically: $G = (V, E)$ where

$$V = \{a, b, c, d, e, f, g\} \text{ and}$$
$$E = \Big\{\{a, b\}, \{a, c\}, \{a, d\}, \{b, d\}, \{c, d\}, \{c, e\}, \{e, f\}, \{e, g\}, \{f, g\}, \{f, h\}\Big\}.$$

Matrix-Style: As a matrix with 1 if there is an edge, and 0 otherwise. (For a directed graph, it may not be symmetric).

$$G = \begin{array}{c|cccccccc} & a & b & c & d & e & f & g & h \\ \hline a & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ b & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ c & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ f & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ g & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ h & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Pictorially: A ball stick model of a graph.



10.2 Clustering on Graphs

So how to cluster a graph? A cluster is a subset $S \subset V$. We are performing top down clustering, so we only need to consider a subset S and its compliment $\bar{S} = V \setminus S$.

In general, we want many edges within a cluster (small width), and few edges between clusters (large split).

- The *volume* of a cluster is $\mathsf{Vol}(S) =$ the number of edges with at least one vertex in V.
- The *cut* between two clusters S, T is $\mathsf{Cut}(S, T) =$ the number of edges with one vertex in S and the other in T.

Then we want a large $\mathsf{Vol}(S)$ for each cluster and a small $\mathsf{Cut}(S, T)$ for each pair of clusters.

Specifically, the *normalized cut* between S and T is $\mathsf{NCut}(S, T) = \frac{\mathsf{Cut}(S,T)}{\mathsf{Vol}(S)} + \frac{\mathsf{Cut}(S,T)}{\mathsf{Vol}(T)}$. And we want to find the cluster S (and compliment $T = V \setminus S$) that has the *minimum* $\mathsf{NCut}(S, T)$. Dividing by $\mathsf{Vol}(S)$ and $\mathsf{Vol}(T)$ prevents us from finding either S or T that is too small, and the $\mathsf{Cut}(S, T)$ on top will force a large split.

For instance, in the above example, the minimum normalized cut is $S = \{a, b, c, d\}$, but the cluster with $S' = \{h\}$ has just as small $\mathsf{Cut}(S', T')$ value. But its normalized cut is $1 + \frac{1}{10} = 1.1$, where as $\mathsf{NCut}(S, T) = \frac{1}{6} + \frac{1}{5} = 0.367$.

10.2.1 Graphs stored in Matrices

Start with an *adjacency* matrix

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

and the *degree* matrix, which along the diagonal stores the degree of each vertex. The *degree* of a vertex is the number of edges that contain that vertex.

$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

10.2.2 (Unnormalized) Laplacian

In some contexts it will be suitable to use the (regular, unnormalized) Laplacian defined:

$$L_0 = D - A = \begin{pmatrix} 3 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & -1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 3 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}.$$

Note that the entries in each row and column of L_0 sum up to 0.

- think of D as the *flow* into a vertex, and
- think of A as the *flow* out of the vertex.

The water keeps flowing, so it does not get stuck anywhere. That is, as much flows in as flows out.

The *eigenvector* of a matrix M is the the vector v such that

$$Mv = \lambda v,$$

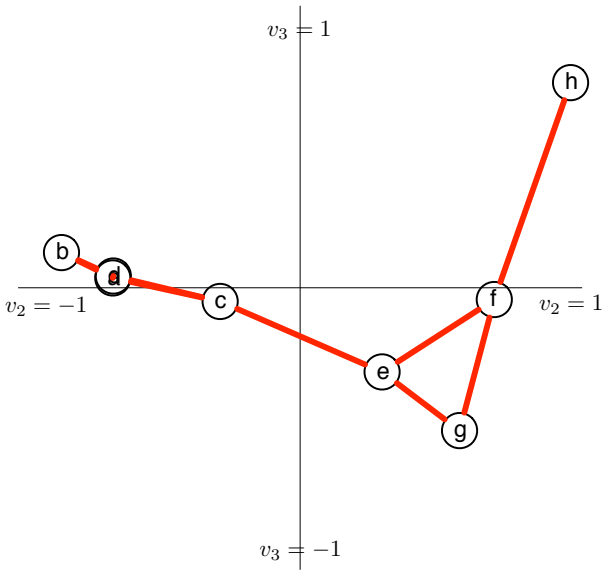
where λ is a scalar. Then λ is the corresponding *eigenvalue*. We usually restrict that $\|v\| = 1$.

There are (typically) several eigenvectors of L_0 (the Laplacian). We list them here sorted by λ.

λ	0	**0.278**	1.11	2.31	3.46	4	4.82
V	$1/\sqrt{8}$	$-.36$	0.08	0.10	0.28	0.25	$1/\sqrt{2}$
	$1/\sqrt{8}$	$-.42$	0.18	0.64	$-.38$	0.25	0
	$1/\sqrt{8}$	$-.20$	$-.11$	0.61	0.03	$-.25$	0
	$1/\sqrt{8}$	$-.36$	0.08	0.10	0.28	0.25	$-1/\sqrt{2}$
	$1/\sqrt{8}$	0.17	$-.37$	0.21	$-.54$	$-.25$	0
	$1/\sqrt{8}$	0.36	$-.08$	$-.10$	$-.28$	0.75	0
	$1/\sqrt{8}$	0.31	$-.51$	$-.36$	$-.56$	0.56	0
	$1/\sqrt{8}$	0.50	0.73	0.08	0.11	0.11	0

This can be calculated easily in matlab using the `[V,Λ] = eig(L)` command.

And here is the drawing of the vertices according to v_2 and v_3, scaled by $1/\sqrt{\lambda_i}$ along each axis. Again the drawing below points a and d are directly on top of each other. From the perspective of the graph, they are indistinguishable. The eigenstructure does not separate them until v_7.



10.2.3 Top-Down Spectral Clustering

The second eigenvector of the normalized Laplacian, called the *Fielder vector*, is a *very important* descriptor of a graph. In the example it is $v_2 = (-.36, -.42, -.20, -.36, 0.17, 0.36, 0.31, 0.50)$ as read off the second column of the above chart.

- It tells us how to best cut the graph.
- It tells us how "best" to put all of the vertices on a single line
- We can set $S = \{v_i \in V \mid u_2(v_i) < 0\}$ and $T = V \setminus S$.

  Then $S = \{a, b, c, d\}$ and $T = \{e, f, g, h\}$.
- Can sometimes do better by checking all possible cuts along v_2 (use any threshold, not only 0). Take one with best $\mathsf{NCut}(S, T)$.

The third eigenvector can be useful too. It can be used (with the second eigenvector) to lay out the vertices in $\mathbb{R}^2$, and can then be used to make a 4-way cut.

[+ +] $S = \{h\}$ defined as $v_2 > 0$ and $v_3 > 0$
[+ −] $T = \{e, f, g\}$ defined as $v_2 > 0$ and $v_3 < 0$
[− +] $U = \{a, b, d\}$ defined as $v_2 < 0$ and $v_3 > 0$
[− −] $R = \{c\}$ defined as $v_2 < 0$ and $v_3 > 0$.

10.2.4 Normalized Laplacian

Before we proceed we will want to use the matrix $D^{-1/2}$; for a diagonal matrix, a power p just takes every element $d_{i,i}$ on the diagonal to that power $d_{i,i}^p$. So

$$D^{-1/2} = \begin{pmatrix} 0.577 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.707 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.577 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.577 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.577 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.577 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.707 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then the normalized *Laplacian* matrix is now the product of $D^{-1/2}$ and A

$$L = I - D^{-1/2}AD^{-1/2} = \begin{pmatrix} 1 & -0.408 & -0.333 & -0.333 & 0 & 0 & 0 & 0 \\ -0.408 & 1 & 0 & -0.408 & 0 & 0 & 0 & 0 \\ -0.333 & 0 & 1 & -0.333 & -0.333 & 0 & 0 & 0 \\ -0.333 & -0.408 & -0.333 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.333 & 0 & 1 & -0.333 & -0.408 & 0 \\ 0 & 0 & 0 & 0 & -0.333 & 1 & -0.408 & -0.577 \\ 0 & 0 & 0 & 0 & -0.408 & -0.408 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.577 & 0 & 1 \end{pmatrix}.$$

We can also convert L_0 to the normalized Laplacian L using the $D^{-1/2}$ matrix as

$$L = I - D^{-1/2}AD^{-1/2} = D^{-1/2}L_0D^{-1/2}.$$

The left- and right-multiplication by $D^{-1/2}$ can be thought of as normalizing by the degrees. That is each entry $P_{i,j}$ of $P = D^{-1/2}AD^{-1/2}$ (and edge (i,j)) is normalized by the amount of flow in and out of the nodes v_i and v_j of its corresponding edge (i,j).

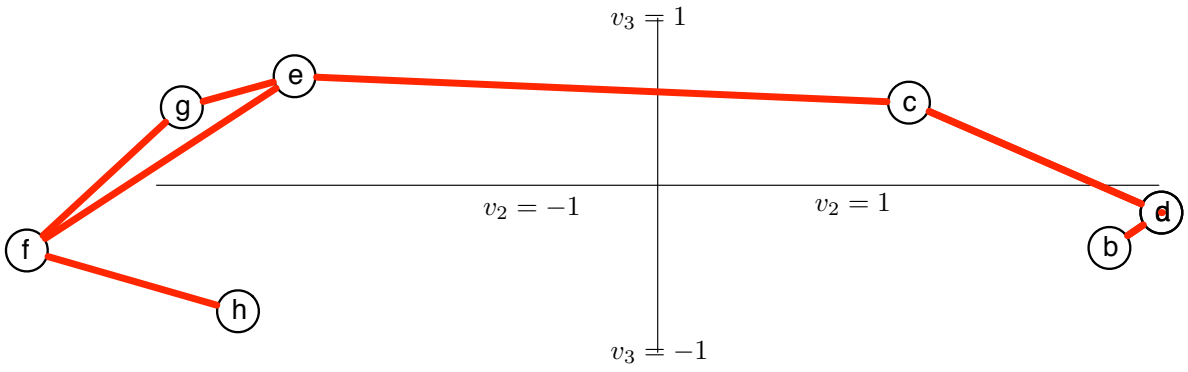
The normalized Laplacian is almost always preferred to the (unnormalized) Laplacian for spectral. This is because the second eigenvalue corresponds to a relaxed solution to the Normalized Cut problem. The cut requires a binary choice, but for an appropriately phrased optimization problem where the normalized cut is the optimal binary solution, the second eigenvalue of the normalized Laplacian is the optimal solution allowing for continuous values.

There are (typically) several eigenvectors of L (the normalized Laplacian): We list them here sorted by λ.

λ	0	**0.125**	0.724	1.00	1.33	1.42	1.66	1.73
V	−.39	0.38	−.09	0.00	0.71	0.26	−.32	0.16
	−.32	0.36	−.27	0.50	0.00	−.51	0.38	−.18
	−.39	0.18	0.36	−.61	0.00	0.03	0.47	−.29
	−.39	0.38	−.09	0.00	−.71	0.26	−.32	0.16
	−.39	−.28	0.48	0.00	0.00	−.57	0.31	0.33
	−.39	−.48	−.29	0.00	0.00	0.05	−.31	−.65
	−.31	−.36	0.27	0.50	0.00	0.51	0.38	−.18
	−.22	−.32	−.61	−.35	0.00	−.07	0.27	0.51

The first eigenvalue of the Laplacian is always 0, up to numerical error.

When drawing the graph using v_2 and v_3 its good to scale the values by $1/\sqrt{\lambda_i}$ along each axis. Note that in the drawing below points a and d are directly on top of each other. From the perspective of the graph, they are indistinguishable. The eigenstructure does not separate them until v_5.



Alternatively, we can use the first d eigenvectors (scaled by eigenvalues) to embed the vertices in $\mathbb{R}^d$. Then we can use any Euclidean clustering algorithm (such as Lloyds for k-means clustering). The smaller the eigenvector, the more important the direction. So the larger the index of the eigenvalue, the smaller the $1/\sqrt{\lambda_i}$ will be. So the top 5 or so (depending on data) may be all required. Notice here how the second eigenvector provides much better separation than the third one.

Affinity matrix. More generally, the adjacency matrix need not be $0-1$. It can be filled with the *similarity* value defined by some similarity function $\mathbf{s} : X \times X \to [0,1]$ between elements of the data set X; then A stands for *affinity*. The diagonal is still diagonal, but is now defined as the sum of elements in a row (or column — it must be symmetric). Then spectral clustering can be run as before. When the similarity of a pair is very small, it is a good heuristic to round the values down to 0 in the matrix to make the algorithm run faster.

This embedding into $\mathbb{R}^2$ is a form of *non-linear dimensionality reduction*. This is popular to use where we set $\mathbf{s}(a,b) = \exp(-\|a-b\|^2/\sigma^2)$ applied to data in $\mathbb{R}^d$ where σ is a scaling parameter that describes in Euclidean distance what is close; and beyond which (say $> 2\sigma$) is far. Applying this with the (unnormalized) Laplacian is called *Laplacian Eigenmaps*.