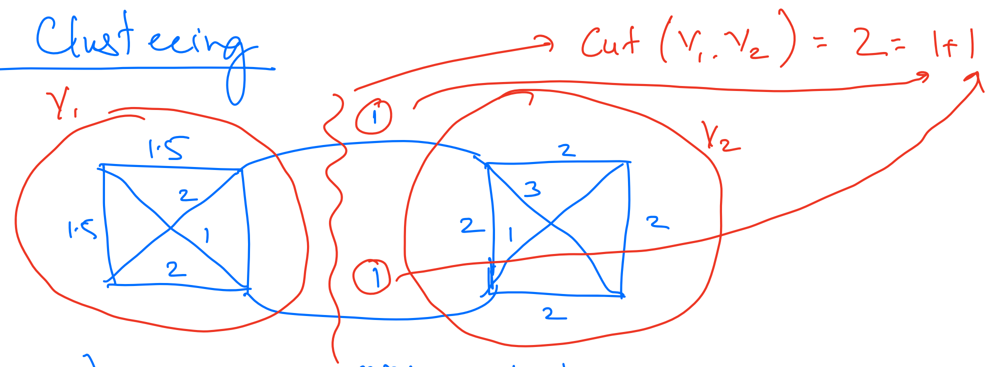
Graph Clustering



$G = (V, E, A)$, $A \in \mathbb{R}^{n \times n}$, $|V| = n$

weighted adjacency matrix of the graph

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

Goal of graph clustering :

Partition V into k clusters of nodes/vertices

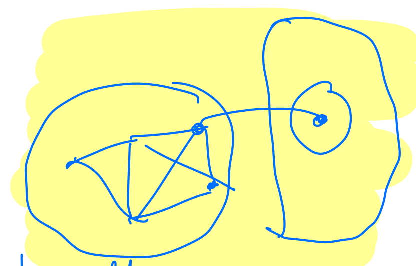
For example, $k=2$

$V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$

Graphs arise in many different applications : social networks, web data, image segmentation, circuit partitioning

$$Cut(V_1, V_2) = \sum_{\substack{v_i \in V_1 \\ v_j \in V_2}} A_{ij}$$

Can try to minimize $Cut(V_1, V_2)$.



Just minimizing $Cut(V_1, V_2)$ does not work well because it leads to extremely unbalanced clusters

One classical way of enforcing balance :

$$\boxed{\begin{aligned} \min_{V_1, V_2} \ & Cut(V_1, V_2) \\ \text{st} \ & |V_1| = |V_2| \end{aligned}} \Bigg\} \text{ NP-hard}$$

Kernighan-Lin heuristic : Starts with some partitioning V_1 & V_2 such that $|V_1| = |V_2|$.

Calculate change in cut if 2 vertices (1 in V_1 and 2nd in V_2) are swapped

In fact, it performs a sequence of swaps ← { Swap the vertices that decreases the cut the most.

Greedy algorithm – more sophisticated variants have been developed (Fiduccia & Mattheyses)

popular for circuit partitioning.

Weighted Graph Cuts

1. Ratio-Cut (G) → $\min_{V_1, V_2} \left(\frac{Cut(V_1, V_2)}{|V_1|} + \frac{Cut(V_1, V_2)}{|V_2|} \right) = Cut(V_1, V_2) \cdot \left(\frac{1}{|V_1|} + \frac{1}{|V_2|} \right)$
$RC(G)$

$$= Cut(V_1, V_2) \cdot \left(\frac{|V_1| + |V_2|}{|V_1||V_2|} \right)$$

$$= \frac{Cut(V_1, V_2) \cdot n}{|V_1||V_2|}$$

2. Normalized Cut (G) → $\min_{V_1, V_2} \left(\frac{Cut(V_1, V_2)}{deg(V_1)} + \frac{Cut(V_1, V_2)}{deg(V_2)} \right)$
$NC(G)$

$$\boxed{deg(V_c) = \sum_{\substack{v_i \in V_c \\ (i,j) \in E}} A_{ij} = links(V_c, V)}$$

$$links(V_c, V_c) = \sum_{\substack{v_i, v_j \in V_c \\ (i,j) \in E}} A_{ij}$$

The above is ~~the~~ only valid for $k = 2$ clusters

Generalize the above definitions of RC and NC to k clusters:

$$\boxed{RC \rightarrow \min_{V_1, V_2, \ldots V_k} \sum_{c=1}^{k} \frac{Cut(V_c, V - V_c)}{|V_c|}}$$

$$\boxed{NC \rightarrow \min_{V_1, V_2, \ldots V_k} \sum_{c=1}^{k} \frac{Cut(V_c, V - V_c)}{deg(V_c)}}$$

$$Cut(V_c, V - V_c) = links(V_c, V - V_c)$$
$$= deg(V_c) - links(V_c, V_c)$$

$$\frac{\deg(V_c) - \text{links}(V_c, V_c)}{\deg(V_c)} = 1 - \frac{\text{links}(V_c, V_c)}{\deg(V_c)}$$

"Normalized Association")

Minimizing "Normalized Cut" is the same as Maximizing "Normalized Association".

<u>Goal</u>: Minimize these weighted graph cut objectives

Spectral Clustering is a (good) heuristic that tries to minimize these objectives — "global heuristic"

1. Associate an indicator vector $y_1, y_2, \ldots, y_k$ for each cluster, $y_i \in \mathbb{R}^n$

$$y_c(j) = \begin{cases} 1, & \text{if } v_j \in V_c \\ 0, & \text{otherwise} \end{cases}$$

$$y_c^T y_c = |V_c|$$

$$y_{c_1}^T y_{c_2} = 0, \quad c_1 \neq c_2$$ (y_{c_1} & y_{c_2} are orthogonal to each other)

2. Define a matrix D: diagonal matrix of degrees ($D \in \mathbb{R}^{n\times n}$)

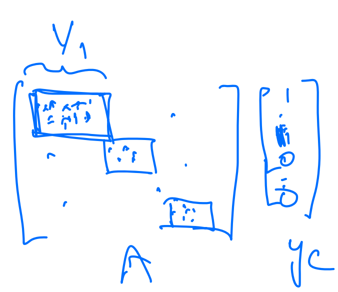
$$D = \begin{bmatrix} \deg(v_1) & & 0 \\ & \deg(v_2) & \\ & & \ddots & \\ 0 & & & \deg(v_n) \end{bmatrix}$$

$$y_c^T D y_c = \deg(v_1) + \cdots + \deg(v_\ell), \quad v_1, v_2, \ldots, v_\ell \in V_c, \ \ell = |V_c|$$

$$\boxed{y_c^T D y_c = \deg(V_c)}$$

$$\boxed{y_c^T A y_c} = \boxed{\text{links}(V_c, V_c)}$$

$\begin{bmatrix} 1 \cdots 1 & 0 \cdots 0 \end{bmatrix}$
y_c^T



$$RC(G) = \sum_{c=1}^{k} \frac{\text{Cut}(V_c, V - V_c)}{|V_c|} = \sum_{c=1}^{k} \frac{\deg(V_c) - \text{links}(V_c, V_c)}{|V_c|}$$

$y_1^T A y_1 = \text{links}(V_1, V_1)$

$$= \sum_{c=1}^{k} \frac{y_c^T D y_c - y_c^T A y_c}{y_c^T y_c}$$

$$\boxed{RC(G) = \sum_{c=1}^{k} \frac{y_c^T (D - A) y_c}{y_c^T y_c} = \sum_{c=1}^{k} \frac{y_c^T L y_c}{y_c^T y_c}}$$

where $L = D - A$ is called the Graph Laplacian

L has various properties :

1. Can show that $L = EE^T$ where E is the node-edge incidence matrix

$$E = \begin{array}{c|c} & \begin{matrix} & (i,j) & \end{matrix} \\ \begin{matrix} i \\ n \\ j \end{matrix} & \begin{bmatrix} & \sqrt{w_{ij}} & \\ & & \\ & \sqrt{w_{ij}} & \end{bmatrix} \\ & |E| \end{array}$$

2. L is a symmetric, positive semi-definite matrix

3. $L\underline{1} = (D - A)\underline{1} = \underline{0}$ ($\underline{1}$ is an eigenvector of L with eigenvalue 0)

4. For any vector x, $x^T L x = x^T E E^T x = \sum_{(i,j) \in E} w_{ij} (x_i - x_j)^2$

$$NC(G) = \sum_{c=1}^{k} \frac{y_c^T (D - A) y_c}{y_c^T D y_c} = \sum_{c=1}^{k} \frac{y_c^T L y_c}{y_c^T D y_c}$$

$$z_c = D^{1/2} y_c \Rightarrow y_c = D^{-1/2} z_c$$

$$NC(G) = \sum_{c=1}^{k} \frac{z_c^T D^{-1/2} L D^{-1/2} z_c}{z_c^T z_c}$$

and $D^{-1/2} L D^{-1/2}$ is called the "Normalized Laplacian"

$\boxed{\text{Spectral Clustering}}$

ratio-cut

$\uparrow$

$Lx = \lambda x$

1. Form the Graph Laplacian $L = D - A$

2. Compute the k smallest eigenvectors (except all 1's vector) $\begin{cases} \text{of } L \text{ (ratio-cut)} \\ \text{of } D^{-1/2} L D^{-1/2} \text{ (NC)} \end{cases}$,

   and assemble them in matrix $Y \in \mathbb{R}^{n \times k}$

$$\text{i-th row} \leftarrow \begin{array}{cc} & \overbrace{\qquad\qquad\qquad}^{k} \\ n & \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_k \\ | & | & & | \end{bmatrix} \end{array} = Y$$

3. Extract clustering from Y

   (run k-means over the n rows of Y)

Spectral Clustering is a powerful "global" heuristic for Graph Clustering.