Clustering nodes in graphs

Why graph clustering is useful?

- Distance matrices are graphs → as useful as any other clustering
- Identification of communities in social networks
- Webpage clustering for better data management of web data

Outline

- k-core decomposition of a graph
- Min s-t cut problem
- Min cut problem
- Spectral graph partitioning

k-core graph decomposition

- Assume an undirected graph G=(V,E)
- The core i of G, denoted by Gi, is a **subgraph** of G such that all nodes in Gi have degree at least i

- The core number of a node u is c(u), if u belongs in the c(u) core but not in core c(u) +1

Min **s-t** cut

- Weighted graph **$G(V,E)$**

- An **s-t** cut **$C = (S,T)$** of a graph **$G = (V, E)$** is a cut partition of **V** into **S** and **T** such that **$s \in S$** and **$t \in T$**

- Cost of a cut: **$Cost(C) = \Sigma_{e(u,v)\ u \in S,\ v \in T}\ w(e)$**

- **Problem:** Given **G**, **s** and **t** find the minimum cost **s-t** cut

Min-cut problem

- Connected, undirected graph **G=(V,E)**
- Assignment of weights to edges: **w: E→R⁺**
- **Cut:** Partition of V into two sets: **V', V-V'**. The set of edges with one end point in **V** and the other in **V'** define the cut
- The removal of the cut disconnects **G**
- **Cost of a cut:** sum of the weights of the edges that have one of their end point in **V'** and the other in **V-V'**

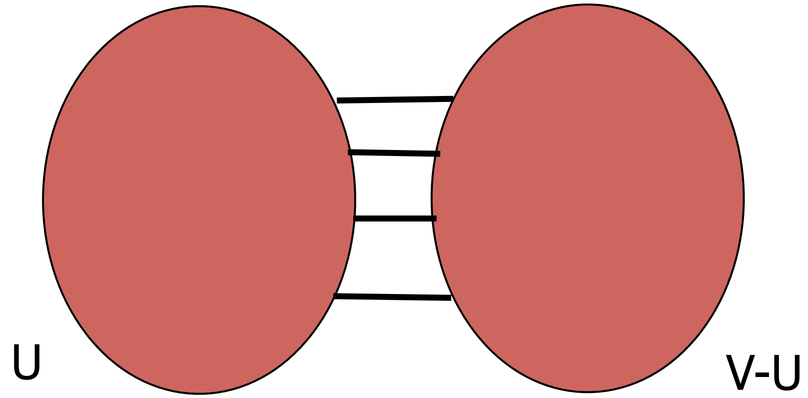
Min cut problem

- Can we solve the min-cut problem using an algorithm for s-t cut?

More on min-cut

- What does it mean that a set of nodes are well or sparsely interconnected?

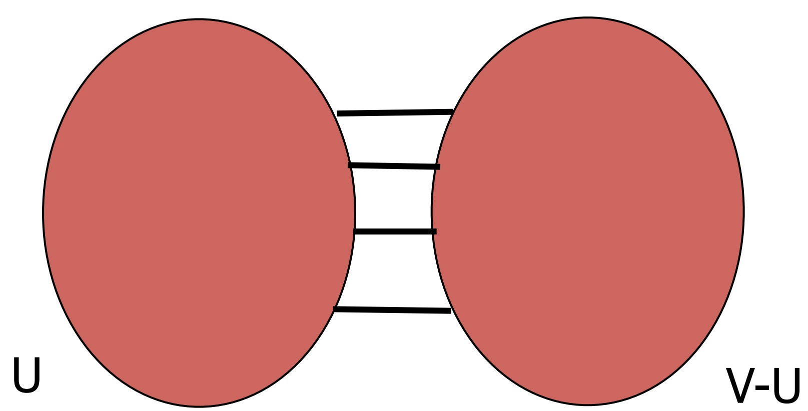
- min-cut: the min number of edges such that when removed cause the graph to become disconnected
  - small min-cut implies sparse connectivity
  - $\min_{U} E(U, V \setminus U) = \sum_{i \in U} \sum_{j \in V \setminus U} A[i,j]$

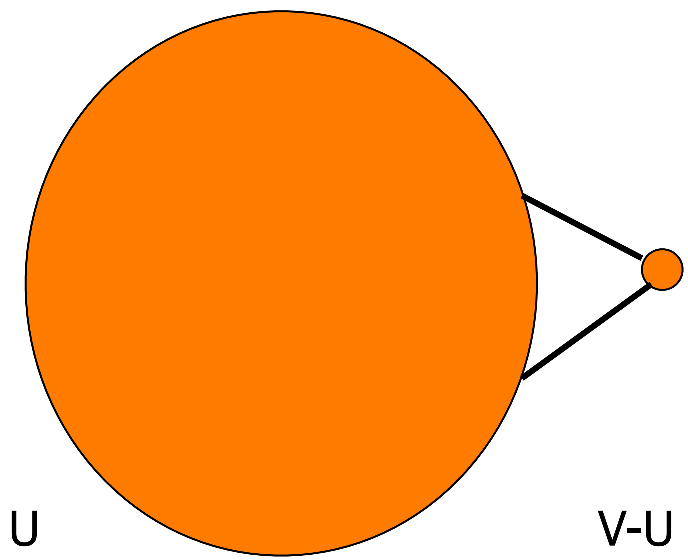


Measuring connectivity

- What does it mean that a set of nodes are well interconnected?

- min-cut: the min number of edges such that when removed cause the graph to become disconnected
  – not always a good idea!





Graph expansion

- Normalize the cut by the size of the smallest component
- Cut ratio:
$$\alpha = \frac{E(U, V \setminus U)}{\min\{|U|, |V \setminus U|\}}$$
- Graph expansion:
$$\alpha(G) = \min_U \frac{E(U, V \setminus U)}{\min\{|U|, |V \setminus U|\}}$$
- We will now see how the graph expansion relates to the eigenvalue of the adjacency matrix A

Spectral analysis

- The Laplacian matrix $L = D – A$ where
  - A = the adjacency matrix
  - $D = diag(d_1,d_2,…,d_n)$
    - d_i = degree of node i
- Therefore
  - $L(i,i) = d_i$
  - $L(i,j) = -1$, if there is an edge (i,j)

Laplacian Matrix properties

- The matrix L is symmetric and positive semi-definite
  - all eigenvalues of L are positive

- The matrix L has 0 as an eigenvalue, and corresponding eigenvector $w_1 = (1,1,\ldots,1)$
  - $\lambda_1 = 0$ is the smallest eigenvalue

The second smallest eigenvalue

- The second smallest eigenvalue (also known as Fielder value) λ_2 satisfies

$$\lambda_2 = \min_{\|x\|=1, x \perp w_1} x^T L x$$

- The vector that minimizes λ_2 is called the Fielder vector. It minimizes

$$\lambda_2 = \min_{x \neq 0} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2} \quad \text{where} \quad \sum_i x_i = 0$$

Spectral ordering

- The values of x minimize

$$\min_{x \neq 0} \frac{\sum_{(i,j)\in E}(x_i - xj)^2}{\sum_i x_i^2} \qquad \sum_i x_i = 0$$

- For weighted matrices

$$\min_{x \neq 0} \frac{\sum_{(i,j)} A[i,j](x_i - xj)^2}{\sum_i x_i^2} \qquad \sum_i x_i = 0$$

- The ordering according to the x_i values will group similar (connected) nodes together

- Physical interpretation: The stable state of springs placed on the edges of the graph

Spectral partition

- Partition the nodes according to the ordering induced by the Fielder vector
- If $u = (u_1,u_2,\ldots,u_n)$ is the Fielder vector, then split nodes according to a value s
  - bisection: s is the median value in u
  - ratio cut: s is the value that minimizes α
  - sign: separate positive and negative values ($s=0$)
  - gap: separate according to the largest gap in the values of u
- This works well (provably for special cases)

Spectral Clustering

a number of variations of the following algorithm

Input: Matrix A in R^{nxn}, number of clusters k

- Compute the Laplacian matrix L from A
- Compute the first k eigenvectors $u_1, u_2, \ldots, u_k$
- Let U in R^{nxk} be the matrix containing the vectors $u_1, u_2, \ldots, u_k$ as columns
- For i=1,…,n let y_i in R^k be the vector of the *ith* row in U
- Cluster the points y_i (i=1,…,n) into k clusters using k-means: $C_1, C_2, \ldots, C_k$

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \text{ in } C_i\}$