We've talked a lot about "clustering" in this class, but it hasn't been very satisfying....

We know that

- Being "highly clustered" is one of our universal properties.
- Clusters are crucial in defining how cascades spread
- Clusters are crucial for community detection.

....

But, our definition of "clustering" feels pretty toy.

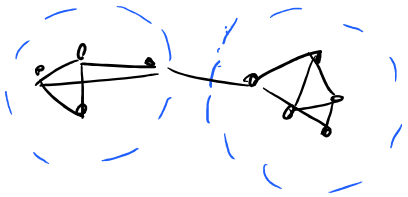
We've basically just counted triangles....
which tells us something, but not much.

The clustering coefficients we've defined are "micro". They tell us about "friends of friends" but not about "communities"

Today our goal is to look at more "macro" definitions of clusters, and to understand how we can find clusters & communities in graphs.

Defining Clusters

Q: What defines a "good" clustering, a.k.a. partioning, of a graph?



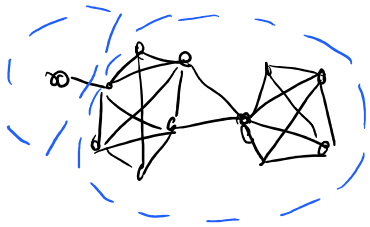
A1: Few edges between the clusters/partitions

Not good enough!

Q: What can go wrong?

A: looking for small cuts means trimming off small, peripheral parts of the graph.

e.g.



Q: What's missing?

A: The clusters should have some "weight" or "volume", i.e., have lots of edges & nodes internally.

Q: So, what's a better measure of a good partition?

A2: We need to balance the cut size with the volume.

Two standard approaches:

hard constraint: All partitions must be large enough, e.g., $\geq (1-\beta)\frac{n}{2}$ (constant; # of nodes).

Soft constraint: Look at the ratio of cut to some notion of volume

(a) # of nodes

(b) # of edges

Soft constraints tend to be prefered because they allow partitions to be sized endogenously.

Four measures

The four most common measures of quality are the following:

- Expansion ($h(S)$)

$$\frac{e(S, \bar{S})}{|S|/n}$$

← size of cut between S & $\bar{S} = V - S$ (vertex set)

- Sparsity ($sp(S)$)

$$\frac{e(S, \bar{S})}{|S|\,|\bar{S}|}$$

- Conductance ($\Phi(S)$)

$$\frac{e(S, \bar{S})}{Vol(S)/n}$$

$Vol(S) = \sum_{i \in S} d_i$ ← degree of i

Note: $\Phi(G) = \min_S \frac{e(S, \bar{S})}{\min(vol(S), vol(\bar{S}))}$ is the conductance of the graph

- Normalized cut ($Ncut(S)$)

$$\frac{e(S, \bar{S})}{Vol(S)} + \frac{e(S, \bar{S})}{vol(\bar{S})}$$

Q: Which do you like?

A: It's hard to say....

Expansion $\approx$ Sparsity (both normalize by node size)

Conductance $\approx$ Normalized Cut (both normalize by edge

Finding partitions

With these definitions in hand, we can now go after the algorithmic question:

"How can we find good partitions of graphs?"

We'll focus on NCut first to be concrete...

Goal: find $S^* = \operatorname{argmin}_S \text{NCut}(S, \bar{S})$

↑

NP-hard!

Finding good partitions is very challenging

So, what now?

... we look for approximations.

It turns out that the key to approximations is the Laplacian of the graph

def: The Laplacian of a graph is

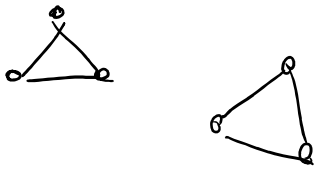
$$L = D - A$$

D: degree matrix, A: adjacency matrix

$$\begin{bmatrix} d_1 & & 0 \\ & d_2 & \\ 0 & & \ddots \\ & & & d_n \end{bmatrix}$$

Q: Why is this connected to partitioning?

A: Consider this simple example...



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

Now lets look at the eigenvectors

$$Lv = \lambda v$$

for $\lambda = 0$ $\quad v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $\quad v_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

exactly the clusters!

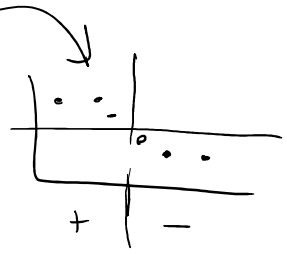
And if we add an edge between we get:

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

which gives

$$\lambda = 0 \qquad v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\lambda = \frac{\sqrt{17}+5}{2} \qquad v_1 = \begin{bmatrix} 1 \\ 1 \\ \frac{\sqrt{17}-3}{2} \\ \frac{-\sqrt{17}-3}{2} \\ -1 \\ -1 \end{bmatrix} \begin{matrix} \\ \\ + \\ \\ - \\ \\ \end{matrix}$$



So, the eigenvector gives us a partition!

.... so, you can "hear" things about the structure of graphs from the Laplacian!

An aside: Can you hear the shape of a drum?

Imagine a drum as simply a shape with a surface over it.

ex:

a surface [illegible]

ex: 

Then it's sound is the frequencies of the vibration → which is characterized by the eigenvalues of the Laplacian!

A fun, classic maths question is:

Can you infer the shape of a drum from hearing it?

i.e. do the eigenvalues of the Laplacian uniquely identify the shape of the drum?

It turns out that the answer is "No"!

have the same spectrum!

Okay, Now lets get back to Ncut.

To see concretely the relationship between Ncut & the Laplacian lets rewrite the

formula a bit...

$$Ncut(S,\bar{S}) = e(S,\bar{S})\left(\frac{1}{vol(S)} + \frac{1}{vol(\bar{S})}\right)$$

let $v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$ with $v_i = \begin{cases} \frac{1}{vol(S)} & i \in S \\ \frac{-1}{vol(\bar{S})} & i \in \bar{S} \end{cases}$

Then $v^T L v = \sum a_{ij}(f_i - f_j)^2 = \sum_{\substack{i\in S \\ j\in\bar{S}}} a_{ij}\left(\frac{1}{vol(S)} + \frac{1}{vol(\bar{S})}\right)^2$

($A = [a_{ij}]$)

& $v^T D v = \sum d_i f_i^2 = \sum_{i\in S} \frac{d_i}{vol(S)^2} + \sum_{i\in\bar{S}} \frac{d_i}{vol(\bar{S})^2}$

$$= \frac{1}{vol(S)} + \frac{1}{vol(\bar{S})}$$

So $Ncut(S,\bar{S}) = \dfrac{v^T L v}{v^T D v}$

This doesn't help us directly, but we can now "relax" the problem and allow v to be free...

$$\arg\min_v \frac{v^T L v}{v^T D v} \quad \text{st} \quad v^T D \underline{1} = 0$$

... which is equivalent to the

2^{nd} eigenvector of a generalized eigenvalue problem:

$$Lv = \lambda Dv$$

Solving this gives v & then we find the approximately optimal partition by taking $i \in S$ iff $v_i \geq 0$.

Q: Why the 2^{nd} eigen value?

A: The first eigenvalue is only useful if the graph is disconnected...
(Recall our examples)

Q: How good is this approximation alg?

A1: In practice, it seems to work very well...
(see slides)

A2: It's a challenge to prove bounds, but its possible.

The key tool is what are called "

Cheeger Bounds"

The classic ones are for conductance, so I'll state those.

Thm: $2\,\underline{\Phi}_G \geq \lambda_2 \geq \underline{\Phi}_G^2/2$

- $\underline{\Phi}_G$: conductance of G
- $\min_S \frac{e(S,\bar{S})}{\min(\mathrm{vol}(S), \mathrm{vol}(\bar{S}))}$
- λ_2: 2nd eigenvalue of Laplacian

So, the eigenvalue gives a bound on how good the conductance is!

$$2\lambda_2 \geq \underline{\Phi}_G \geq \lambda_2/2$$

Connections to random walks & Pagerank

At this point we have good measures for partitions & a way to find partitions that are approximately optimal...

What's left?

Q: can we find good partitions more quickly?

Remember, pagerank is an eigenvalue problem, but we don't solve it by finding an eigenvector directly, we find it by iterating through a random walk. (our random websurfer).

Q: Can we do the same thing for finding partitions?

A: Yes!

What is the transition matrix for the random walk on this graph?

$$P = [d_{ij}] \quad d_{ij} = \begin{cases} \frac{1}{d_i} & \text{if } (i,j) \in E \\ 0 & \text{else} \end{cases}$$

$$P = D^{-1}A$$

Remember $L = D - A$.

So, $A = D - L$.

$$\Rightarrow P = D^{-1}(D - L) = I - D^{-1}L$$

So, the results can be ported to random walks!

random walks!

(... and conductance here determines the convergence rate of our iterative approach for pagerank!)

A cheegers inequality here is the following:

Thm: $2\Phi_G \geq \lambda_2 \geq \beta_G^2/\delta \log n \geq \Phi_G^2/\delta \log n$

β_G: minimum conductance over a random walk of k steps from every vertex (for appropriate k)

Intuition: Do a random walk from every vertex (by iterating the matrix) and if nodes are in the same cluster the random walks will connect them!

This can be much faster especially in graphs from complex networks (which tend to have heavy-tailed spectral gaps).

Wapping up

That's basically what I have to say about spectral partitioning...

Two final remarks

1) We've talked about 2-partitions. To do k partitions you just look at more eigenvalues/eigenvectors... But, if you don't know k, you're in trouble. Spectral techniques are not always great at determining k... though looking at the full spectrum can sometimes let you eyeball it.

2) There are many other styles of algorithms too!

→ Local improvement algorithms
Start w/ a partition and then make "local improvement" steps to find a near-optimal final partition.

→ Flow-based methods
Use multicommodity flow to find bottlenecks and then cut those edges

Some sense of when to use which approaches:...

→ Spectral methods fail on "stringy" graphs

→ Flow methods fail on "expander graphs" (most complex networks have large expansion)

→ Local improvement methods often get stuck in local extrema, and should only be used as "cleanup" methods, i.e., as ways to refine approximate solutions given by other approaches.