CSCI 4022 Fall 2021
Community Detection

The **AGM model**: generates the links of a network *probabilistically*, using the community probabilities p_C.

1. For each pair of nodes u, v in a community $A \in C$, we connect them with probability p_A.
2. ... but u and v might also share e.g. community $B \in C$, which would *independently* connect them with probability p_B.

   $P(A \sim B)$: edge from 0 p OR from 1 q

3. In the final network u and v will be connected by an edge if an edge is *generated* from *any one* of their shared communities

What's the overall probability that *at least one* such edge is drawn?

0	1	Prob
T	T	$p \cdot q$
T	F	$p(1-q)$
F	T	$(1-p)q$
F	F	$(1-p)(1-q)$

Instead! $P(\text{no } A\text{-}B) = P(\text{not from any community they share})$.

Announcement and Reminders

+ Z office hour 5p – 6p

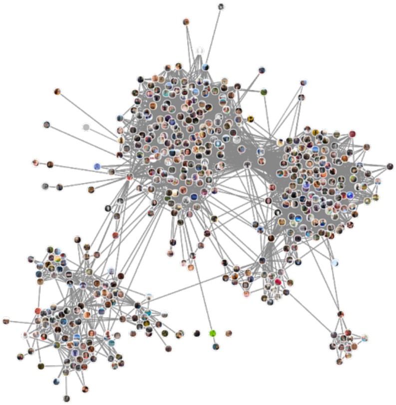
1. Hw 5 due tonight HW6 **and** project proposals for next Monday.

Networks and Communities

Examples:

Facebook friends.

1. Nodes: users
2. Edges: friendships



Networks and Communities

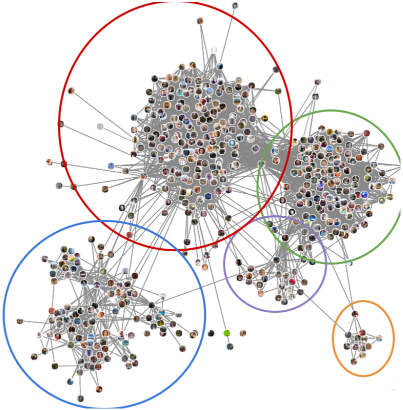
Examples:

Facebook friends.

1. Nodes: users
2. Edges: friendships

This also invites the notion of a community:

1. red: College friends
2. green: high school friends
3. blue: graduate school friends
4. purple: family
5. orange: summer internship friends

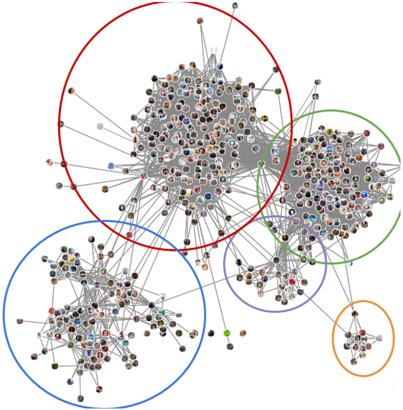


Networks and Communities

Examples:

Facebook friends.

1. Nodes: users
2. Edges: friendships



What could we do with this?

Describe points with *similar* community affiliations: recommend friends, events, or advertisements.

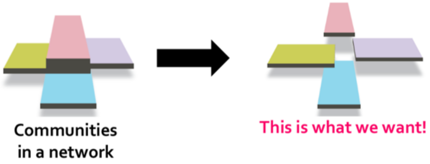
Unlike clusters, these groups **overlap**, and nodes will belong to multiple groups.

Communities

Goal: Find a way to describe the communities in a graph.

Similarity to clustering:

1. There is overlap between different "clusters" – each is a community
2. Unlike clustering, can belong to multiple communities
3. The goal: can we identify these social network communities?



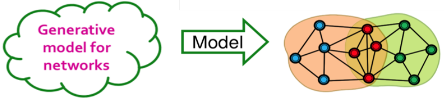
Modeling a Network and Communities

If we can come up with a **model**, we could make a graph. Think of this as the conditional problem: *given* communities, how might we draw a graph?

Goal: given a model, generate networks

1. Given some nodes. . .
2. The model will have a set of parameters that govern the connections among nodes
3. We will estimate these parameters (~~next~~ today time). But if we have a generative model that's based on *probability*, than the "best choice of parameters," sounds in all *likelihood* to be a problem we've dealt with before...

Question: Given a set of nodes, how will our model generate the edges of the network?



P_0 1, P_2

The Community-Affiliation Graph Model (AGM)

Model: Given a set of nodes, what's a reasonable way to *generate* edges of a network?

Comm.

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ A & 1 & 1 & 1 & 0 \\ B & 1 & 0 & 1 & 1 \end{array}$$

$P(A \overset{\text{edge}}{-} B \text{ exists}) = 1 - \underbrace{(1-P_0)}_{\text{not from } 0}\underbrace{(1-P_2)}_{\text{not from } 2}$

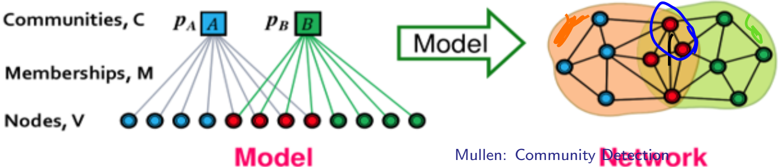
The **AGM model**: $B(V, C, M, \{p_c\})$

1. V = the set of nodes.
2. C = the set of communities.
3. M = the set of memberships (think of it as edges from V to C, **or** object that's `len(V)` $\times$ `len(C)` and holds a score if row "person" is inside column "community".)
4. Each community C gets a unique probability p_C denoting the probability of C generating a connection between members of C.

$p(\text{edge}) = 1 - [(1-p)(1-q)]$

$p(\text{not edge}) = (1-p)\cdot(1-q)$

→ term for each community shared



The Community-Affiliation Graph Model (AGM)

The **AGM model**: generates the links of the network *probabilistically*, using the community probabilities p_C.

1. For each pair of nodes u, v in a community $A \in C$, we connect them with probability p_A.
2. ... but u and v might also share e.g. community $B \in C$, which would *independently* connect them with probability p_B.
3. In the final network u and v will be connected by an edge if an edge is *generated* from *any one* of their shared communities

$\neg$ (one or more) = exactly 0.

What's the overall probability that *at least one* such edge is drawn? **DeMorgan's Laws**

$P(\text{at least one edge}) = 1 - P(\text{no edges})$...

$= 1 - P(\text{no edge from first shared comm AND no edge from first shared comm AND...})$

Result: under AGM, the probability of the edge (u, v) is

$$P(u, v) = 1 - \prod_{C \in M_u \cap M_v} (1 - p_c).$$

shared communities

The Community-Affiliation Graph Model (AGM)

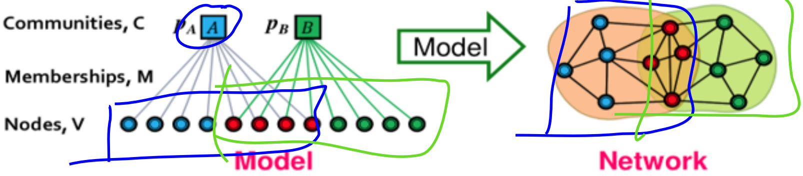
The **AGM model**: generates the links of the network *probabilistically*, using the community probabilities p_C.

Result: under AGM, the probability of the edge (u, v) is

$$P(u, v) = 1 - \prod_{C \in M_u \cap M_v} (1 - p_c).$$

where M_u is the set of all communities of member u.
We may also include a *background probability*, where

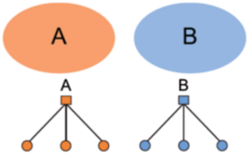
$P(u, v) = \varepsilon$ if $M_u \cap M_v = \emptyset$ for some small ε. What does this ε represent?



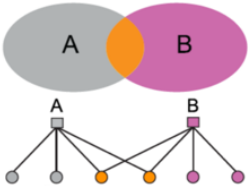
Flexibility of AGM

We can use AGM to express a variety of different common structures:

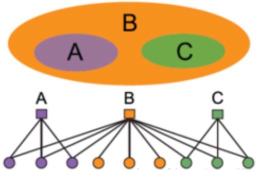
Non-overlapping; disjoint



Overlapping



Nested (Subset) Communities

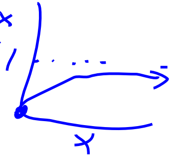


Now, to actually estimate those probabilities *given* a graph...

Modifying AGM

F people $\rightarrow$ $\overset{\text{comm}}{\begin{bmatrix} 3 & 0 & 1 & 15 \\ 4 & 0 & 0 & 8 \end{bmatrix}}$

$1 - e^{-x}$



For two reasons, we're going to add a tweak to the AGM model...

1. Membership in communities is not binary (0/1), it has a strength. (The team captain is more likely to form within-team connections than a supporting member)

   Define: $F_{u,A}$: the membership strength of node u in community a. Set $F_{u,a} = 0$ if and only if u is absolutely **not** a member of A.

2. We adjust the resulting probability to link two nodes in the same community A to be

   For each community $\quad 1 - e^{(-.3 \cdot 4)}$

$$\boxed{P_A(u,v)} = 1 - \exp\left(-F_{u,A} \times F_{v,A}\right).$$

   Why this? If u and v share *many* groups, the probability of the edge (u, v) given by

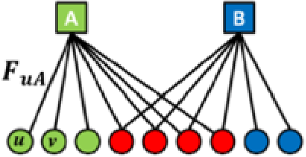
$$P(u,v) = 1 - \boxed{\prod_{C \in M_u \cap M_v} (1 - p_c)}.$$

   not from any

   is going to be **much easier** if it simplifies!

BigCLAM

The resulting model is called the Cluster Affiliation Model for Big Networks, or BigCLAM.



$F_{u,A}$: the membership strength of node u in community a. Set $F_{u,a} = 0$ if and only if u is absolutely **not** a member of A.

$$P_A(u,v) = 1 - \exp\left(-F_{u,A} \times F_{v,A}\right).$$

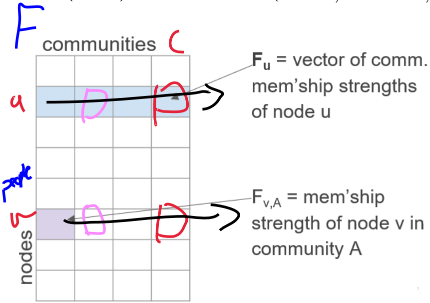
Goal: Find the k communities inside a *given* graph... kind of like a GMM or k-means!

Prob (Graph | F)

BigCLAM

So we've replaced the p_c's of AGM with a new probability-per-community that relies on membership strengths:

$$P_A(u,v) = 1 - \exp\left(-F_{u,A} \times F_{v,A}\right).$$



…This helps with the probability that nodes u and v share *at least one edge* through any community:

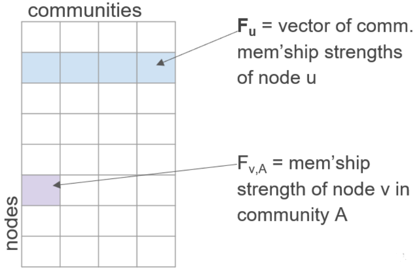
$$P(u,v) = 1 - \prod_c \left(1 - \underbrace{P_c(u,v)}_{1 - e^{-(F_{u,c} \cdot F_{v,c})}}\right)$$

BigCLAM

$e^a \cdot e^b \cdot e^c = e^{a+b+c}$

So we've replaced the p_c's of AGM with a new probability-per-community that relies on membership strengths:

$$P_A(u,v) = 1 - \exp\left(-F_{u,A} \times F_{v,A}\right).$$



…This helps with the probability that nodes u and v share *at least one edge* through any community:

$$P(u,v) = 1 - \prod_c \left(1 - P_c(u,v)\right)$$

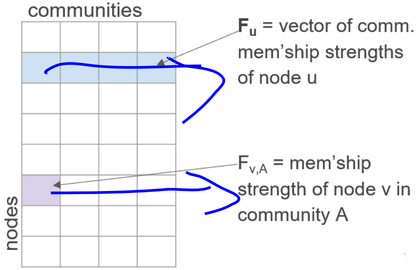
$$= 1 - \prod_c \left(1 \cancel{- (1} - \exp\left(-F_{u,c} \times F_{v,c}\right)\cancel{)}\right)$$

$$= 1 - \prod_c \exp\left(-F_{u,c} \times F_{v,c}\right)$$

BigCLAM

So we've replaced the p_c's of AGM with a new probability-per-community that relies on membership strengths:

$$P_A(u,v) = 1 - \exp\left(-F_{u,A} \times F_{v,A}\right).$$



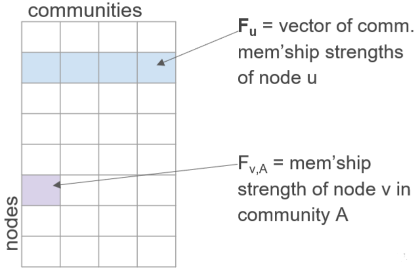
...This helps with the probability that nodes u and v share *at least one edge* through any community:

$$\begin{aligned} P(u,v) &= 1 - \prod_c (1 - P_c(u,v)) \\ &= 1 - \prod_c (1 - (1 - \exp(-F_{u,c} \times F_{v,c}))) \\ &= 1 - \prod_c \exp(-F_{u,c} \times F_{v,c}) \\ &= 1 - \exp\left(\sum_c -F_{u,c} \times F_{v,c}\right) \end{aligned}$$

BigCLAM

So we've replaced the p_c's of AGM with a new probability-per-community that relies on membership strengths:

$$P_A(u,v) = 1 - \exp\left(-F_{u,A} \times F_{v,A}\right).$$



…This helps with the probability that nodes u and v share *at least one edge* through any community:

1 - p(not edge)

$$\begin{aligned}
P(u,v) &= 1 - \prod_c \left(1 - P_c(u,v)\right) \\
&= 1 - \prod_c \left(1 - \left(1 - \exp\left(-F_{u,c} \times F_{v,c}\right)\right)\right) \\
&= 1 - \prod_c \exp\left(-F_{u,c} \times F_{v,c}\right) \\
&= 1 - \exp\left(\sum_c -F_{u,c} \times F_{v,c}\right) \\
&= 1 - \exp\left(-\boldsymbol{F_u} \cdot \boldsymbol{F_v}\right)
\end{aligned}$$

not from C

BigCLAM Probabilities

$P(\cdot)^{u} \quad P(\cdot) \quad {}^{v}\,P(\cdot)^{w}$

Example: Suppose for node set $V = \{u, v, w\}$ and communities $\{A, B, C, D\}$, we have the following membership strength matrix F. What are the probabilities of a connection between u and v? Between u and w? v and w?

F=

	A	B	C	D
u	0	1.2	0	0.2
v	0.5	0	0	0.8
w	0	1.8	1	0

$P(edge_{u,v}) = 1 - e^{-(F_u \cdot F_v)}$

$p(u,v) = 1 - e^{(-F_u F_v)} = 1 - e^{-(.2 \cdot .8)} = 1 - e^{-.16}$

$p(u,w) = 1 - e^{-(0 \cdot 0 + 1.2 \cdot 1.8 + 0 \cdot 1 + .2 \cdot 0)} = 1 - e^{(1.2 \cdot 1.8)}$

$p(v,w) = 1 - e^{-0} = 1 - 1 = 0$

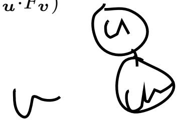
BigCLAM Probabilities

Example: Suppose for node set $V = \{u, v, w\}$ and communities $\{A, B, C, D\}$, we have the following membership strength matrix F. What are the probabilities of a connection between u and v? Between u and w? v and w?

F=

	A	B	C	D
u	0	1.2	0	0.2
v	0.5	0	0	0.8
w	0	1.8	1	0

$$P(edge_{u,v}) = 1 - e^{-(F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}$$



Solution:

$$P(u, v) = 1 - e^{-(F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} = 1 - e^{-0.16} = 0.14$$

$$P(u, w) = 1 - e^{-(F_{\boldsymbol{u}} \cdot F_{\boldsymbol{w}})} = 1 - e^{-2.16} = 0.88$$

$$P(v, w) = 1 - e^{-(F_{\boldsymbol{v}} \cdot F_{\boldsymbol{w}})} = 0$$

BigCLAM Implementation

So connection probabilities are pretty easy, as $P(u, v) = 1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})$ ...As long as we know F. Given the undirected graph $G(V, E)$, how do we find F?

BigCLAM Implementation

So connection probabilities are pretty easy, as $P(u, v) = 1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})$ ...As long as we know F. Given the undirected graph $G(V, E)$, how do we find F?

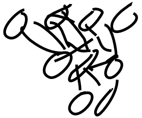
Answer: Pick F to maximize the *likelihood function*. Recall that a likelihood function is the probability of the data *given* the model, or the probability that we observed exactly which edges were in the data set and which ones weren't.

1000 edges

Com 1

$\emptyset$

400 edges out of $\binom{50}{2}$ possible

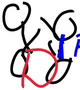


50

25

Com 2

100 edges out of $\binom{25}{2} = 25 \cdot 12 = 300$



p(edge)

p(non-edge)

BigCLAM Implementation

So connection probabilities are pretty easy, as $P(u,v) = 1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})$ ...As long as we know F. Given the undirected graph $G(V,E)$, how do we find F?

Answer: Pick F to maximize the *likelihood function*. Recall that a likelihood function is the probability of the data *given* the model, or the probability that we observed exactly which edges were in the data set and which ones weren't.

The edge connections between nodes are a Bernoulli process:

- $(u,v) \in E$ with probability $\boxed{P(u,v)}$ or $(u,v) \notin E$ with probability $1 - P(u,v)$
  not
- Each edge connection is established probabilistically, independently of the others.

Result: Probability of a given graph (edge set E) being established, given F, is:

$$L(F) = \boxed{\prod_{(u,v)\in E} P(u,v)} \prod_{(u,v)\notin E} (1 - P(u,v))$$

(?) things

BigCLAM Likelihood

Result: Probability of a given graph (edge set E) being established, given F, is:

$$L(F) = \prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))$$

Goal: find the community affiliations F that maximize this value.

Here's the thing about likelihood functions: nobody likes them. Nobody. He's an unstable numerical mess.

Likelihood Function's little sister, the log-likelihood function, on the other hand, everybody likes:

$1-e^{-(u\cdot v)}$ $\qquad$ $1-(1-e^{-u\cdot v})$

$$l(F) = \log\left(\prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))\right)$$

$$l(F) = \sum_{(u,v)\in E} (\log\left[1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right]) - \sum_{(u,v)\notin E} (\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}))$$

BigCLAM Log-Likelihood

Goal: find the community affiliations F that maximize the log-likelihood function:

$$l(F) = \log\left(\prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))\right)$$

$$l(F) = \sum_{(u,v)\in E} \left(\log\left[1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right]\right) + \sum_{(u,v)\notin E} \log\left(1 - 1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right)$$

$$l(F) = \underbrace{\sum_{(u,v)\in E} \left(\log\left[1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right]\right)}_{\text{edges}} - \underbrace{\sum_{(u,v)\notin E} F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}}_{\text{no-edges}}$$

now... **commence maximization**!!

BigCLAM Log-Likelihood

Goal: estimate F by optimizing the log-likelihood function of

$$l(F) = \sum_{(u,v)\in E} \log\left(1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right) - \sum_{(u,v)\notin E} F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}$$

via *gradient ascent*.

Denote $\boxed{N(u)}$ by the set of neighbor nodes of u, which are connected to u by edges but may or may not share many communities. **Idea:**

1. The log-likelihood $l(F)$ defines a surface with respect to the "coordinates" (rows of F)
2. We want to ascend to the top (maximum) of the log-likelihood surface
3. Fun fact: the gradient points uphill
4. Given a particular guess for a row in F, $F_{\boldsymbol{u}}$, we can improve it be taking a step in the direction of the gradient of $l(F)$ with respect to (vector) $F_{\boldsymbol{u}}$: $\nabla l(F_{\boldsymbol{u}})$

Gradient Ascent/Descent

The idea of following the derivative to find the (local) maximum or minimum value of a function is called gradient ascent or gradient descent. It requires:

1. The ability to calculate the slope of the function we're trying to min/max
2. An idea of how large of steps to take; a step size or *learning rate*

$$\underbrace{F^{(k+1)}}_{\text{new guess}} = \underbrace{F^{(k)}}_{\text{old guess}} + \underbrace{\nu}_{\text{step size}} \underbrace{F'(z^{(k)})}_{\text{step direction}}$$

In our case, we're going to *update* our estimates for F by taking small steps in the direction of the community affiliations for node u. In other words: A step is an update to u to be more closely affiliated with it's neighbors. Then repeat for *every* node u.

The *gradient* is the multivariate direction we're supposed to take steps in!

Gradient Ascent/Descent

So we need a derivative to determine which way to take a step.
Idea: move community affiliations of a node closer to the affiliations of its neighbors.

In practice, we're differentiating

$$l(F) = \sum_{(u,v)\in E} \log\left(1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right) - \sum_{(u,v)\notin E} F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}$$

but we'll go at it *one specific location* at a time, so we're looking at

$$l(F_u) = \sum_{v\in N(u)} \log\left(1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right) - \sum_{v\notin N(u)} F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}$$

and differentiating with respect to row u
(In other words "how should we update our knowledge of person u").
Calculus friends: $\frac{d}{dx}\log(1 - f(x)) =$,
$\frac{d}{dx}e^{f(x)} =$

Gradient Ascent/Descent

So we need a derivative to determine which way to take a step.
Idea: move community affiliations of a node closer to the affiliations of its neighbors.

In practice, we're differentiating

$$l(F) = \sum_{(u,v)\in E} \log\left(1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right) - \sum_{(u,v)\notin E} F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}$$

but we'll go at it *one specific location* at a time, so we're looking at

$$l(F_u) = \sum_{v\in N(u)} \log\left(1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right) - \sum_{v\notin N(u)} F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}$$

and differentiating with respect to row u
(In other words "how should we update our knowledge of person u").
Calculus friends: $\frac{d}{dx}\log(1-f(x)) = \frac{f'(x)}{1-f(x)}$,
$\frac{d}{dx}e^{f(x)} = f'(x)e^{f(x)}$

The BigCLAM gradient

$$\nabla l(F_u) = \frac{d}{dF_{\boldsymbol{u}}} \sum_{v \in N(u)} \log\left(1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right) - \sum_{v \notin N(u)} F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}$$

Each term in the first sum is a derivative of $\log\left(1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})\right)$, which gives $F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}$.

Each term in the second sum is a derivative of $F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}}$, so we are left with just $F_{\boldsymbol{v}}$.

Result:

$$\nabla l(F_u) = \left\langle \underbrace{\sum_{v \in N(u)} F_{v,A} \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v},A}}_{\nabla_A l(F_u)}, \ldots \right\rangle$$

The BigCLAM gradient

The full update:

$$\nabla l(F_u) = \langle \sum_{v \in N(u)} F_{v,A} \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v},A},$$
$$\sum_{v \in N(u)} F_{v,B} \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v},B},$$
$$\sum_{v \in N(u)} F_{v,C} \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v},C},$$
$$\ldots, \rangle$$

Or: *for each community*, the corresponding entry to the vector $\nabla l(F_u)$ is the one that pushes u closer to the communities in it's neighbor set $N(u)$ and further from the communities not in its neighbor set.

The BigCLAM Iteration

The full update:

$$\nabla l(F_u) = \langle \sum_{v \in N(u)} F_{v,A} \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v},A},$$

$$\sum_{v \in N(u)} F_{v,B} \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v},B},$$

$$\ldots, \rangle$$

Or **Iterate:**

1. Compute gradient of $l(F)$ with respect to (vector) Fu: $\nabla l(F_u)$ (keeping others fixed)
2. Update the row F_u as: $F_u^{new} = F_u^{old} + \nu \cdot \nabla l(F_u)$. ($\nu$ is a step size (usually small))
3. If any component c of F_u is negative $(F_{u,c} < 0)$, reset $F_{u,c} = 0$. (*Reflect:* why might this happen?)

The BigCLAM Iteration

1. Compute gradient of $l(F)$ with respect to (vector) Fu: $\nabla l(F_u)$ (keeping others fixed)
2. Update the row F_u as: $F_u^{new} = F_u^{old} + \nu \cdot \nabla l(F_u)$.
3. If any component c of F_u is negative $(F_{u,c} < 0)$, reset $F_{u,c} = 0$.

As written, this happens to be pretty slow! We can spruce it up a little, though! The steps in vector shorthand:

$$\nabla l(F_u) = \sum_{v \in N(u)} F_v \frac{\exp(-\boldsymbol{F_u} \cdot \boldsymbol{F_v})}{1 - \exp(-\boldsymbol{F_u} \cdot \boldsymbol{F_v})} - \sum_{v \notin N(u)} \boldsymbol{F_v}$$

Cleanup: F is sparse, since $N(u)$ is usually much smaller than all nodes. This means most of the additions are in the $\sum_{v \notin N(u)}$ sum. But we could rewrite:

$$\sum_{v \notin N(u)} \boldsymbol{F_v} = \sum_{v} \boldsymbol{F_v} - F_u - \sum_{v \in N(u)} \boldsymbol{F_v}$$

The BigCLAM

$$
\begin{aligned}
\nabla l(F_u) &= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v}} \\
&= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \left(\sum_{v} F_{\boldsymbol{v}} - F_u - \sum_{v \in N(u)} F_{\boldsymbol{v}} \right)
\end{aligned}
$$

The BigCLAM

$$
\begin{aligned}
\nabla l(F_u) &= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v}} \\
&= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \left(\sum_{v} F_{\boldsymbol{v}} - F_u - \sum_{v \in N(u)} F_{\boldsymbol{v}} \right) \\
&= \sum_{v \in N(u)} F_v \left(\frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} + 1 \right) + F_u - \sum_{v} F_{\boldsymbol{v}}
\end{aligned}
$$

The BigCLAM

$$
\begin{aligned}
\nabla l(F_u) &= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v}} \\
&= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \left(\sum_{v} F_{\boldsymbol{v}} - F_u - \sum_{v \in N(u)} F_{\boldsymbol{v}} \right) \\
&= \sum_{v \in N(u)} F_v \left(\frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} + 1 \right) + F_u - \sum_{v} F_{\boldsymbol{v}} \\
&= \sum_{v \in N(u)} F_v \left(\frac{1}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} \right) + F_u - \sum_{v} F_{\boldsymbol{v}}
\end{aligned}
$$

The BigCLAM

$$
\begin{aligned}
\nabla l(F_u) &= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \sum_{v \notin N(u)} F_{\boldsymbol{v}} \\
&= \sum_{v \in N(u)} F_v \frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} - \left(\sum_v F_{\boldsymbol{v}} - F_u - \sum_{v \in N(u)} F_{\boldsymbol{v}} \right) \\
&= \sum_{v \in N(u)} F_v \left(\frac{\exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} + 1 \right) + F_u - \sum_v F_{\boldsymbol{v}} \\
&= \sum_{v \in N(u)} F_v \left(\frac{1}{1 - \exp(-F_{\boldsymbol{u}} \cdot F_{\boldsymbol{v}})} \right) + F_u - \sum_v F_{\boldsymbol{v}}
\end{aligned}
$$

What did we win?? Original RH sum: $v \notin N(u)$ was linear in total # of nodes. Now we have just $|N(u)|$ size updates! We can also cache/re-use the sum-over-people community scores in $\sum_v F_{\boldsymbol{v}}$!

Acknowledgments

We will implement BigCLAM in a course notebook. But there are a couple of major concerns with the algorithm that we'll touch on to open next time:

1. How do we initialize F for our gradient ascent?
2. How might we choose k?

Next time: More on graphs: *cuts* and *partitions*, too!

Some material is adapted/adopted from Mining of Massive Data Sets, by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University) `http://www.mmds.org`