

# CSCI 4022 Spring 2021

## Mullen: Hubs and Authorities

### Announcements:

1. Exam grading not quite done.
2. Work on both HW5 and consider project data sets

## Announcement and Reminders

Let's talk **Projects**! The end goal of your project is the following components.

1. A write-up, around 5-8 pages including figures. Should include a short literature review – where did your data come from, what are similar methods or results on similar problem – as well as a full description of both numerical methods and results.
2. Compiled code in a Jupyter notebook that can replicate your results and any figures in 1). You *must* use at least one CSCI4022 method, although comparisons to tools you've used in other classes are encouraged.
3. A short talk, around 5 minutes per group, where you can describe and provide visualizations of your results.

The first part of the evaluation (10%) is a proposal (Due 25 Oct), worth 10 pts. It should be a few paragraphs ( $\approx 500$  words?) and should describe a *problem* you're interested in or a data set you wish to explore, and at least a suggestion of methods you might apply to it. There should be very little math: what's the data going to look like, what's your idea of interesting conclusions that might come out of analyzing it.

## PageRank: Power Iteration

GOAL: find vector  $r$  so that  $Ar = r$    
  $Ar = \lambda r$

1. Suppose that there are  $N$  total web pages to rank.
2. Initialize:  $r^{(0)} = [1/N, 1/N, \dots, 1/N]$
3. Iterate:  $r^{(t+1)} = Mr^{(t)}$
4. Stop if:  $d(r^{(t+1)}, r^{(t)})$  is small. A typical stop would be  $d(r^{(t+1)}, r^{(t)}) < \varepsilon$  under appropriate norm (like  $L_1$ , so  $d(r^{(t+1)}, r^{(t)}) = \sum_{i=1}^N |r_i^{(t+1)} - r_i^{(t)}|$  for some small  $\varepsilon$ )

Implementation note: this finds the eigenvector corresponding to the largest eigenvalue. Which is 1 for a stochastic matrix, and *uniquely* one if the underlying Markov chain is ergodic (well-connected). For a matrix with multiple spider-traps, the eigenvalue will be higher multiplicity!

## Full PageRank:

**The Google solution:** a blend of the the solution to spider traps and dead ends.

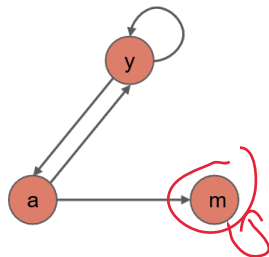
1. With probability  $\beta$ , follow a real link at random.
2. With probability  $1 - \beta$ , *teleport* or jump to any random page, equally likely
3. Common values for  $\beta$ : 0.8-0.9
4. Brin-Page, 1998 (with tens of thousands of citations!) gives the updates PageRank formula:

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

*(Handwritten annotations: A red box around the first term, a blue circle around  $\beta$ , and a blue circle around  $(1 - \beta)$  in the second term.)*

The resulting transition matrix is given by

$$A = \beta M + (1 - \beta) \overbrace{\begin{bmatrix} 1 \\ N \end{bmatrix}}^{\text{every entry is } 1/N}_{N \times N}$$



# Computing Page Rank

In practice, there are a few non-obvious implementation tricks.

1. Our data is usually saved *by column* (out-links) in a sparse matrix, where an update to  $r_j^{new}$  would ordinarily use the row  $j$  of  $M$  of  $A$ . This is ok, as an update is actually just a sum over all in-links, so we can update our sums by column as long as just increment  $r$ -values.
2. There are two options for normalization, necessary in the case of dead ends.
  - 2.1 Update  $r$  with  $r = r/|r|$  after performing all updates.
  - 2.2 Update  $r$  by adding in  $(1 - |r|) / N$  to all  $N$  entries.

Do you believe these are equivalent? Does it depend on anything?

## Topic-Specific Computations

We give a subset  $S$  of the pages greater weight in topic-specific PageRank.

$$\text{Old: } A_{ij} = \beta M_{ij} + \frac{1 - \beta}{N}$$

$$\text{New: } A_{ij} = \begin{cases} \beta M_{ij} + \frac{1 - \beta}{|S|} & \text{if } i \in S \\ \beta M_{ij} + 0 & \text{else} \end{cases}$$

1. We gave all pages in the teleport set  $S$  equal weight ( $1/|S|$ )
2. and gave pages not in the teleport set 0 weight
3. We could have also assigned *different* weights to different pages

Computation is the same as usual:  $r_{\text{new}} = M \cdot r_{\text{old}} + \text{constant}$ , only the addition is now only nonzero for pages in the teleport set.

This solution also can help combat link spamming.

## Link Spamming: The Math

Last time: a spam farm of size  $M$  gets pagerank  $y$  based on it's "valid" pagerank  $x$ .

$$y = \frac{x}{1 - \beta^2} + C \frac{M}{N}$$

if there is a self-link


1. Since  $x$  was the non-farmed rank of the page  $t$ , we call  $1/(1 - \beta^2)$  the *multiplier* effect of the farm. For example,  $\beta = 0.85$  leads to a multiplier of 3.6.
2. By making  $M$  large, we can also make  $y$  nearly as large as we want!
  - ▶  $N$  is enormous though, so  $M$  won't ever truly dominate the internet as a whole...
  - ▶ but it doesn't have to: it only has to dominate similar results!
3.  $\frac{1}{1 - \beta^2}$  gets larger as  $\beta \rightarrow 1$ . This is because our random walker can get *completely stuck* in the spam farm: it's a giant spider trap! OTOH,  $C = \frac{\beta}{(1 + \beta)} \rightarrow 1/2$ , but this term can grow as  $M$  grows as well, because the spam farm starts to own more of the teleport set than it should, since it's  $M/N$  proportion of the internet as a whole.

## TrustRank: Who to Trust?

**TrustRank**: topic-specific PageRank where the teleport set = set of *trusted* pages. The challenge is picking the *right set* of **seed pages** or **trusted** pages to be the teleport set.

Designating a smaller set to be the authorities on a topic is very similar to local search variants of PageRank.

**TrustRank**: teleport set has the *trust*

- ▶ we propagate that trustworthiness throughout the network:  
run a topic-specific PageRank where the teleport set = trusted pages set
- 
 Implementation note: *→, in for fixed steps* we can actually start with all trust = 1 (benefit of the doubt)...  
 The resulting PageRanks are the TrustRanks (slightly different b/c won't sum to 1 if initially all pages have trust = 1)
- ▶ pages closer to trusted set have more trust, farther away has less



## Qualifying Spam

If our goal is to describe whether or not a page is spam, we need a statistic or measure for it!

1. **Option 1:** Pick a threshold TrustRank value; all pages below the threshold marked as spam.  
This risks marking new pages as spam though, among other false positive type errors
2. **Option 2:** With TrustRank, we start with trusted pages and propagate that trust  
Turn it around: Can we estimate the fraction of a page's PageRank comes from spam pages?

## Qualifying Spam

If our goal is to describe whether or not a page is spam, we need a statistic or measure for it!

1. **Option 1:** Pick a threshold TrustRank value; all pages below the threshold marked as spam.

This risks marking new pages as spam though, among other false positive type errors

2. **Option 2:** With TrustRank, we start with trusted pages and propagate that trust  
Turn it around: Can we estimate the fraction of a page's PageRank comes from spam pages?

Define:

$r_p$  = PageRank of page  $p$ , and  $r_p^+$  = PageRank with teleport into **trusted pages only**  $r_p^- =$  PageRank of page  $p$  from "spam" pages  $r_p^- = r_p - r_p^+$

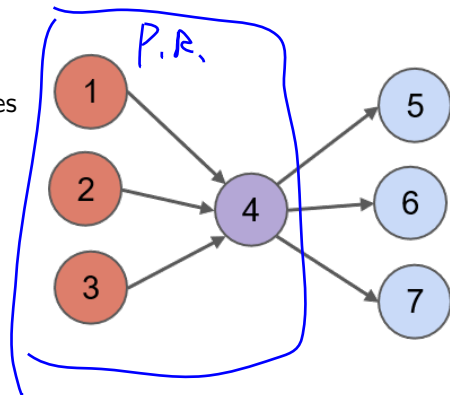
**Definition:** The *spam mass* of page  $p$  is  $\frac{r_p^-}{r_p}$  Pages with sufficiently high spam mass are marked as spam

## What about the out-links?

PageRank: use in-links as votes to determine which websites are most relevant to a search query.

If lots of “important” websites link to yours, you are also important. Nice!

Topic-Specific PageRank: what is your PageRank if the teleport set is restricted to the topic of interest?



**Consider the following:** Suppose my web page links to a bunch of spammy, terrible websites.

## What about the out-links?

PageRank: use in-links as votes to determine which websites are most relevant to a search query.

If lots of “important” websites link to yours, you are also important. Nice!

Topic-Specific PageRank: what is your PageRank if the teleport set is restricted to the topic of interest?



**Consider the following:** Suppose my web page links to a bunch of spammy, terrible websites.

- ▶ I claim to be linking to useful information about the history of pants...
- ▶ ... but when you follow the links, they only lead to websites selling cheap slacks.
- ▶ Question 1: How does this affect your opinion and the relevance of my website?
- ▶ Question 2: Does PageRank capture this?

# Hubs and Authorities

Similarly to PageRank, we use links as votes

But in contrast to PageRank, we look at both **incoming and outgoing** links

**The idea:** if we are interested in movies, then ...

Of course, finding pages that are all about movies is useful. We call these **authorities** (TS-PageRank!) But it is also useful to find pages that link to movies... **hubs**

Each page has **two scores**: **Hub score** = quality as an expert; does your page link to “good” pages?

**Authority score** = quality as content; do “good” pages link to your page?

Note that PageRank is focused on authority score.



## Hubs and Authorities

So we have **hubs**, that tell you where to go to find information about a topic. **Examples:**

1. List of index of newspapers
2. Course bulletin
3. List of US auto manufacturers  
*u.wiki.pedia?*

And then we have **authorities**, the best sites to find useful information about the topic.

**Examples:**

1. Newspaper page itself
2. Course home page
3. Auto manufacturer home page

Each page has **both scores**... and they affect each other!

## HITS informally

The basic algorithm we use for computing the hub and authority scores is *hyperlink-induced topic selection* (HITS).

- ▶ Yields a measure of importance of pages/documents, similar to PageRank.
- ▶ Was developed around the same time, but met a very different fate.

**Algorithm:** Initialize vectors for all  $N$  nodes' authority scores  $a$  and hub scores  $h$ . Iterate until convergence:

1. For each node  $i$ , update the authority score as the sum of the incoming hub scores
2. For each node  $i$ , update the hub scores as the sum of the outgoing authority scores
3. For each node  $i$ , normalize so  $h$  and  $a$ ... (choose one:)

3.1 have largest component = 1

3.2 have magnitude 1

3.3 have sum of components = 1 (like PageRank)

# HITS formally

Let  $L :=$  an *adjacency of link* matrix for the pages. This is like the link matrix  $M$  from PageRank, but not stochastic. Instead;  $L_{ij} = 1$  if  $i \rightarrow j$ ; else  $L_{ij} = 0$ .

**Algorithm:** Initialize, then iterate until convergence:

1. Update  $\mathbf{h}$  via  $\mathbf{h}_i^{(t+1)} = \sum_{i \rightarrow j} \mathbf{a}_j^{(t)} = \sum_j L_{ij} \mathbf{a}_j^{(t)}$ . This reduces to the matrix-vector multiplication  $\mathbf{h}^{(t+1)} = L \mathbf{a}^{(t)}$ . Then normalize  $\mathbf{h}$ .
2. Update  $\mathbf{a}$  via  $\mathbf{a}_i^{(t+1)} = \sum_{j \rightarrow i} L_{ij} \mathbf{h}_j^{(t)} = \sum_j L_{ji} \mathbf{h}_j^{(t)}$ . This reduces to the matrix-vector multiplication  $\mathbf{a}^{(t+1)} = L^T \mathbf{h}^{(t)}$ . Then normalize  $\mathbf{a}$ .

Idea: scan a row for all the outlinks of a page when we update  $\mathbf{h}$ , then scan the corresponding column using the transpose to update  $\mathbf{a}$ .



# HITS Convergence

As always, we need a measure for when to stop iterating.

1. For an established  $\varepsilon$ , we can check whether *both*  $d(\mathbf{h}^{(t+1)}, \mathbf{h}^{(t)})$  **and**  $d(\mathbf{a}^{(t+1)}, \mathbf{a}^{(t)})$  are less than  $\varepsilon$ . If so, we're converged and can break out.
2. We can use any reasonable measure of distance between two vectors in  $\mathbb{R}^n$ :

2.1 Euclidean ( $L_2$ -norm):  $d(\mathbf{h}^{(t+1)}, \mathbf{h}^{(t)})^2 = \sum_i \left( h_i^{(t+1)} - h_i^{(t)} \right)^2$

2.2  $L_1$ -norm:  $d(\mathbf{h}^{(t+1)}, \mathbf{h}^{(t)}) = \sum_i \left| h_i^{(t+1)} - h_i^{(t)} \right|$

# HITS and Eigen-stuff

**Algorithm:** Initialize  $\mathbf{h}^{(0)}$ ,  $\mathbf{a}^{(0)}$  as vectors of all 1s.

Iterate until convergence:

1. Update  $\mathbf{a}^{(t+1)} = L^T \mathbf{h}^{(t)}$

2. Normalize  $\mathbf{a}^{(t+1)}$

3. Update  $\mathbf{h}^{(t+1)} = L \mathbf{a}^{(t)}$ .

4. Normalize  $\mathbf{h}$ .

full step:  $\mathbf{h} = L \mathbf{a}^{(t)}$   $\mathbf{h} = L L^T \mathbf{h}$

**Theory:** As in power iteration for PageRank, HITS will converge to **fixed points**  $\mathbf{h}^*$ ,  $\mathbf{a}^*$ . In other words, if  $\mathbf{h}$  is a fixed point, then after normalizing  $\mathbf{h}$  should be a *fixed* multiple of  $L \mathbf{a}$ .

$$\mathbf{a} = \frac{L^T \mathbf{h}}{\|L^T \mathbf{h}\|} = \frac{L^T L \mathbf{a}}{\|L^T L \mathbf{a}\|}$$

## HITS and Eigen-stuff

**Algorithm:** Initialize  $\mathbf{h}^{(0)}$ ,  $\mathbf{a}^{(0)}$  as vectors of all 1s.

Iterate until convergence:

1. Update  $\mathbf{a}^{(t+1)} = L^T \mathbf{h}^{(t)}$
2. Normalize  $\mathbf{a}^{(t+1)}$
3. Update  $\mathbf{h}^{(t+1)} = L \mathbf{a}^{(t)}$ .
4. Normalize  $\mathbf{h}$ .

**Theory:** As in power iteration for PageRank, HITS will converge to **fixed points**  $\mathbf{h}^*$ ,  $\mathbf{a}^*$ . In other words, if  $\mathbf{h}$  is a fixed point, then after normalizing  $\mathbf{h}$  should be a *fixed* multiple of  $L\mathbf{a}$ . But this simplifies, since  $\mathbf{a}$  must also be some multiple  $\mu$  of  $L^T \mathbf{h}$  :

# HITS and Eigen-stuff

$$a = L^T h \cdot \#$$

**Algorithm:** Initialize  $h^{(0)}$ ,  $a^{(0)}$  as vectors of all 1s.

→ Converge

Iterate until convergence:

1. Update  $a^{(t+1)} = L^T h^{(t)}$
2. Normalize  $a^{(t+1)}$
3. Update  $h^{(t+1)} = La^{(t)}$ .
4. Normalize  $h$ .

$$h = L a \cdot \# = h$$

$$h = 1 \cdot L a$$

$$= 1 \cdot L L^T h$$

**Theory:** As in power iteration for PageRank, HITS will converge to **fixed points**  $h^*$ ,  $a^*$ . In other words, if  $h$  is a fixed point, then after normalizing  $h$  should be a *fixed* multiple of  $La$ . But this simplifies, since  $a$  must also be some multiple  $\mu$  of  $L^T h$ :

$$h = \lambda La = \lambda L (\mu L^T h) = \lambda \mu L L^T h \quad \text{and} \quad a = \mu L^T h = \mu L^T (\lambda La) = \lambda \mu L^T La$$

**Result:**  $h^*$  and  $a^*$  are the principal eigenvectors of  $LL^T$  and  $L^T L$ .

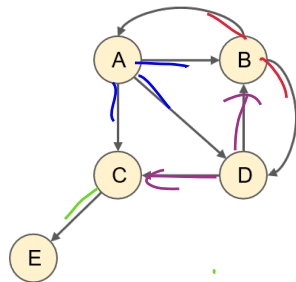
## HITS, Example

Use max-component = 1 normalization and the  $L_2$ -norm to diagnose convergence of the HITS algorithm to determine the hub and authority scores for the network shown here.

Note that by default,  $L$  is rows for out-links, the opposite of the matrix  $M$  from PageRank.

$L =$

	A	B	C	D	E
A	1	1	1	0	0
B	1	0	0	1	0
C	0	0	0	1	1
D	0	1	0	1	0
E	0	0	0	0	0

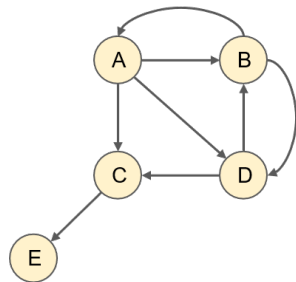


## HITS, Example

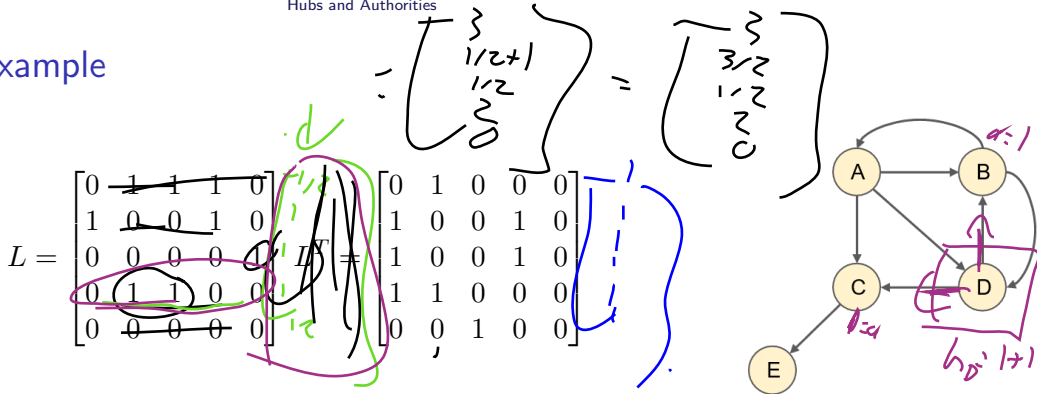
Use max-component = 1 normalization and the  $L_2$ -norm to diagnose convergence of the HITS algorithm to determine the hub and authority scores for the network shown here.

Note that by default,  $L$  is rows for out-links, the opposite of the matrix  $M$  from PageRank.

$$L = \begin{matrix} & \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix} L^T = \begin{matrix} & \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$$



## HITS, Example



**Update 1a:**  $\mathbf{a}^{(1)} = L^T \mathbf{h}^{(0)} \implies \mathbf{a}^{(1)} = [1, 2, 2, 2, 1]$ .

**Normalize:** to make the max of 1, divide all by 2.  $\mathbf{a}^{(1)} = [1/2, 1, 1, 1, 1/2]$

**Update 1h:**  $\mathbf{h}^{(1)} = L \mathbf{a}^{(1)} \implies \mathbf{h}^{(1)} = [3, 3/2, 1/2, 2, 0]$ .

**Normalize:** to make the max of 1, divide all by 3.  $\mathbf{h}^{(1)} = [1, 1/2, 1/6, 2/3, 0]$

## Directed Graphs, Underview



We'll implement HITS in notebook 13.

Both PageRank and HITS solve the problem of quantifying the value of a *link* from a page  $u$  to a page  $v$ .

**PageRank:** the value of the link depends on the value of pages linking to  $u$

**HITS:** the value also depends on the value of pages  $u$  links to

That will conclude our discussion of quantifying directed graphs. For an undirected graph though,  $L$  and  $L^T$  would be the same matrix, since all edges would be symmetric! We could still do a random walk on such a graph to measure its central locations. But we also might want to explore some new methods!



## Networks and Communities

We have been looking at the connections between **web pages**

We represented the internet as a network, with an associated graph (nodes, edges)

Topic-specific PageRank – examines the rankings only within a page's specific area of interest

Result: all the pages within that area of interest are a sort of community

The internet, and life in general, is organized into networks, clusters and communities

...how do we detect and model this structure? (and exploit it for profit and glory)

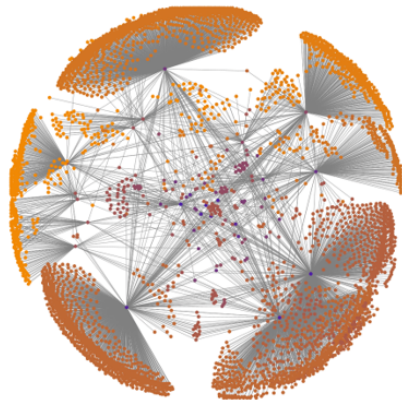


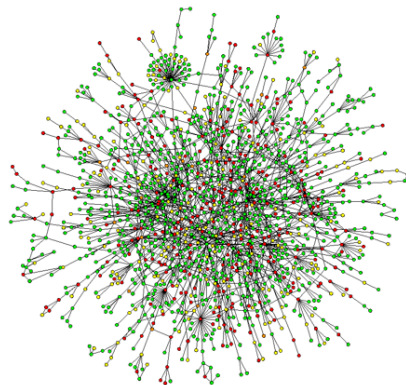
Image credit: Dunne and Shneiderman (2013)

# Networks and Communities

## Examples:

Protein-protein interactions:

1. Nodes: proteins
2. Edges: physical interactions

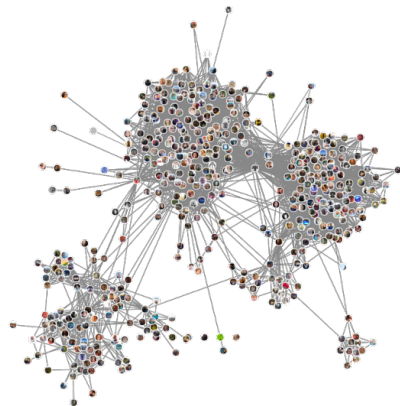


# Networks and Communities

## Examples:

Facebook friends.

1. Nodes: users
2. Edges: friendships



# Networks and Communities

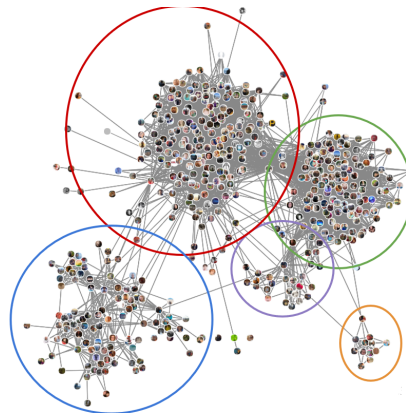
## Examples:

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

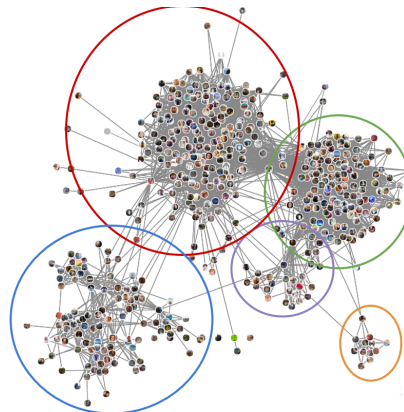


# Networks and Communities

## 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.

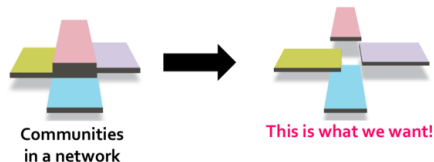
Unlike clusters, these groups **overlap**, and nodes will belong to multiple groups.

# 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 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...

*pledge / properties of users, etc.*

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

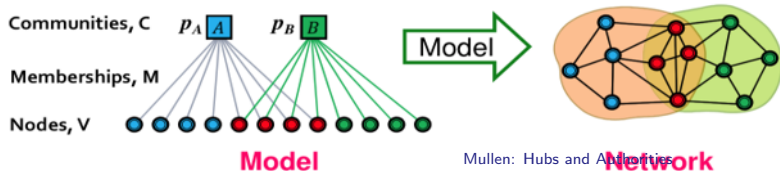


# The Community-Affiliation Graph Model (AGM)

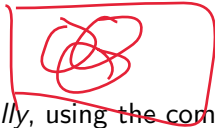
**Model:** Given a set of nodes, what's a reasonable way to *generate* edges of a network?

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  $\text{len}(V) \times \text{len}(C)$  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  $C$ .







## 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

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

they share  $A, B, C$  w/  $p_A, p_B, p_C$   
 $p(\text{edge from } A \mid \text{edge from } B \mid \text{edge from } C).$

## 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

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

## 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

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

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

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

## 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

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

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

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

## The Community-Affiliation Graph Model (AGM)

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

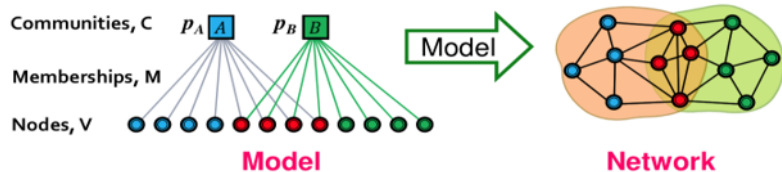
**Result:** under AGM, the probability of the edge  $(u, v)$  is

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

where  $M_u$  is the set of all communities of member  $u$ .

We may also include a *background probability*, where

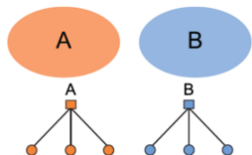
$P(u, v) = \varepsilon$  if  $M_u \cap M_v = \emptyset$  for some small  $\varepsilon$ . What does this  $\varepsilon$  represent?



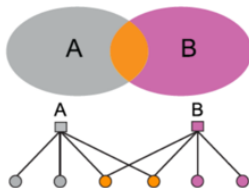
# Flexibility of AGM

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

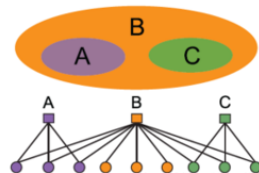
## Non-overlapping; disjoint



## Overlapping



## Nested (Subset) Communities

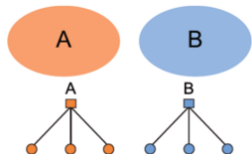


Next time we'll dive into actually estimating those probabilities *given* a graph.

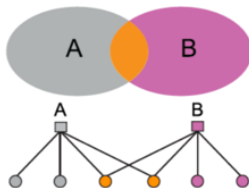
# Flexibility of AGM

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

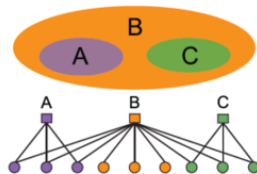
**Non-overlapping; disjoint**



**Overlapping**



**Nested (Subset) Communities**



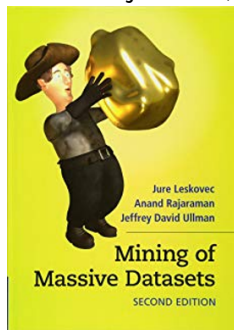
Next time we'll dive into actually estimating those probabilities *given* a graph.

Recall: **Maximum Likelihood** is the common mathematical tool for estimating *parameters* of a model if we can state what the probability of a graph *given* its parameters is!

## Acknowledgments

Next time: More on graphs: *undirected* graph methods!

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



Special thanks to Tony Wong for sharing his original adaptation and adoption of slide material.