

Mining Social-Network Graphs

Analysis of Large Graphs:
Overlapping Communities

New Topic: Graph Data

High dim.
data

Locality
sensitive
hashing

Clustering

Dimensional
ity
reduction

Graph
data

PageRank,
SimRank

Community
Detection

Spam
Detection

Infinite
data

Filtering
data
streams

Web
advertising

Queries on
streams

Machine
learning

SVM

Decision
Trees

Perceptron,
kNN

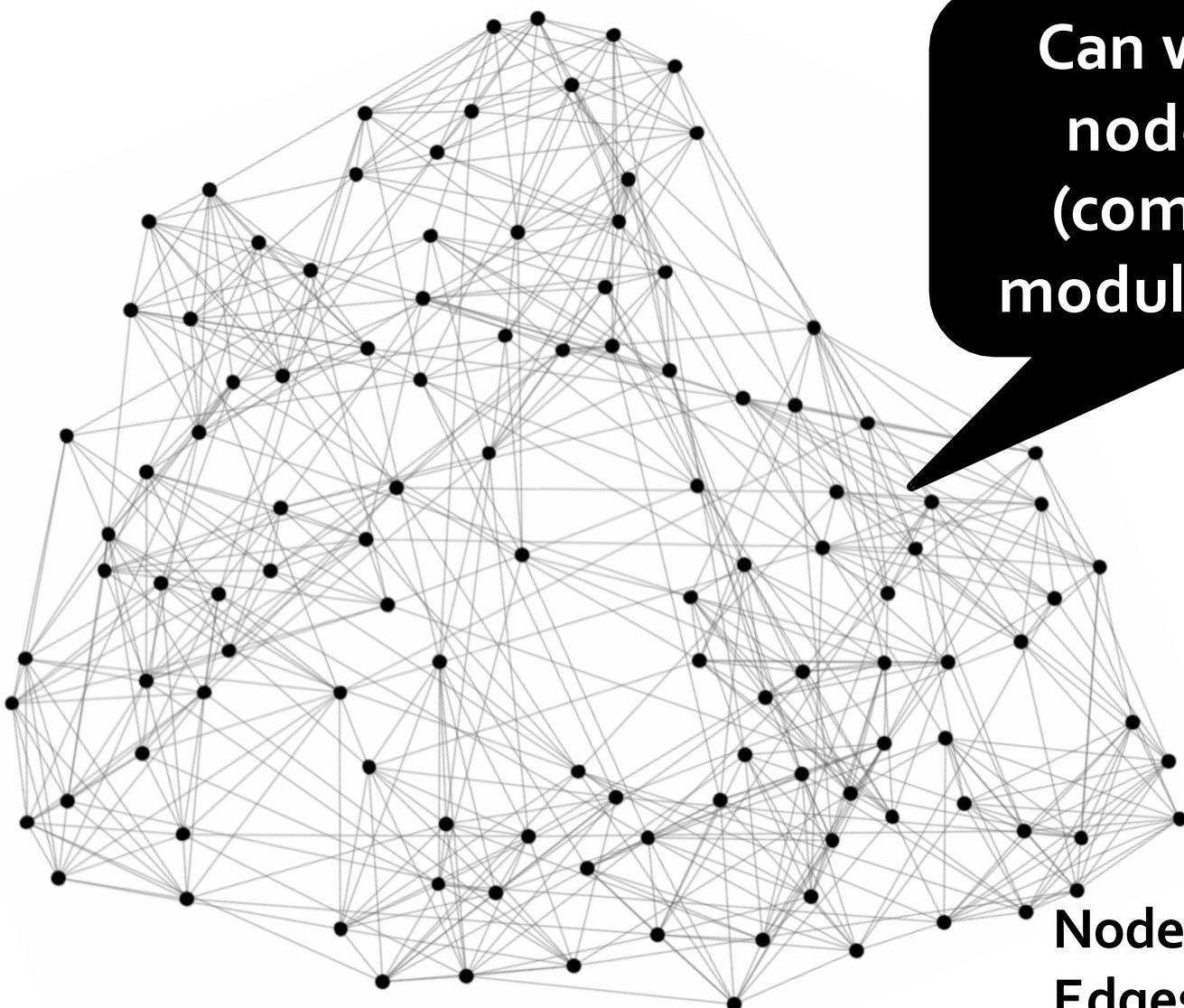
Apps

Recommen
der systems

Association
Rules

Duplicate
document
detection

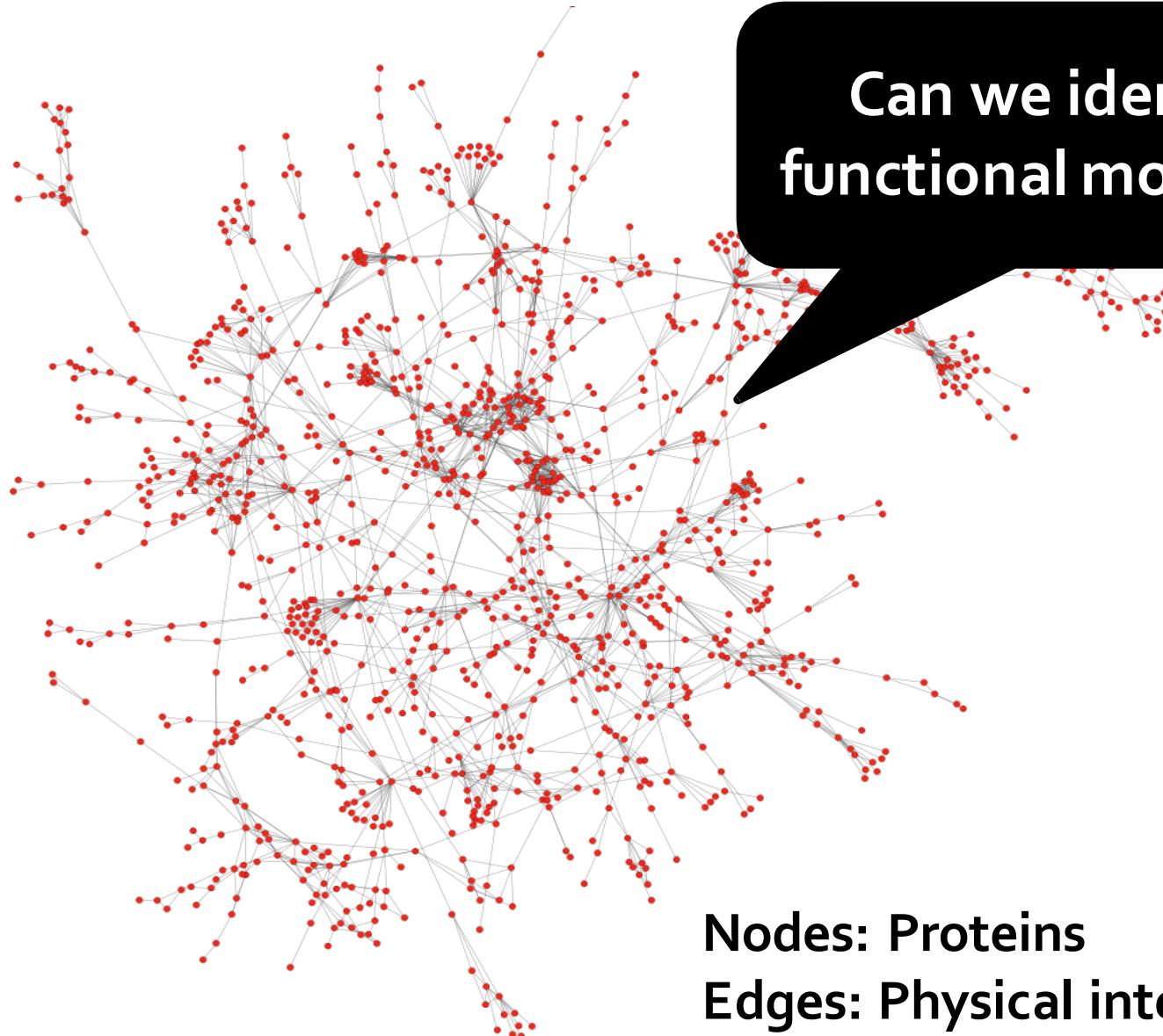
Identifying Communities



Can we identify
node groups?
(communities,
modules, clusters)

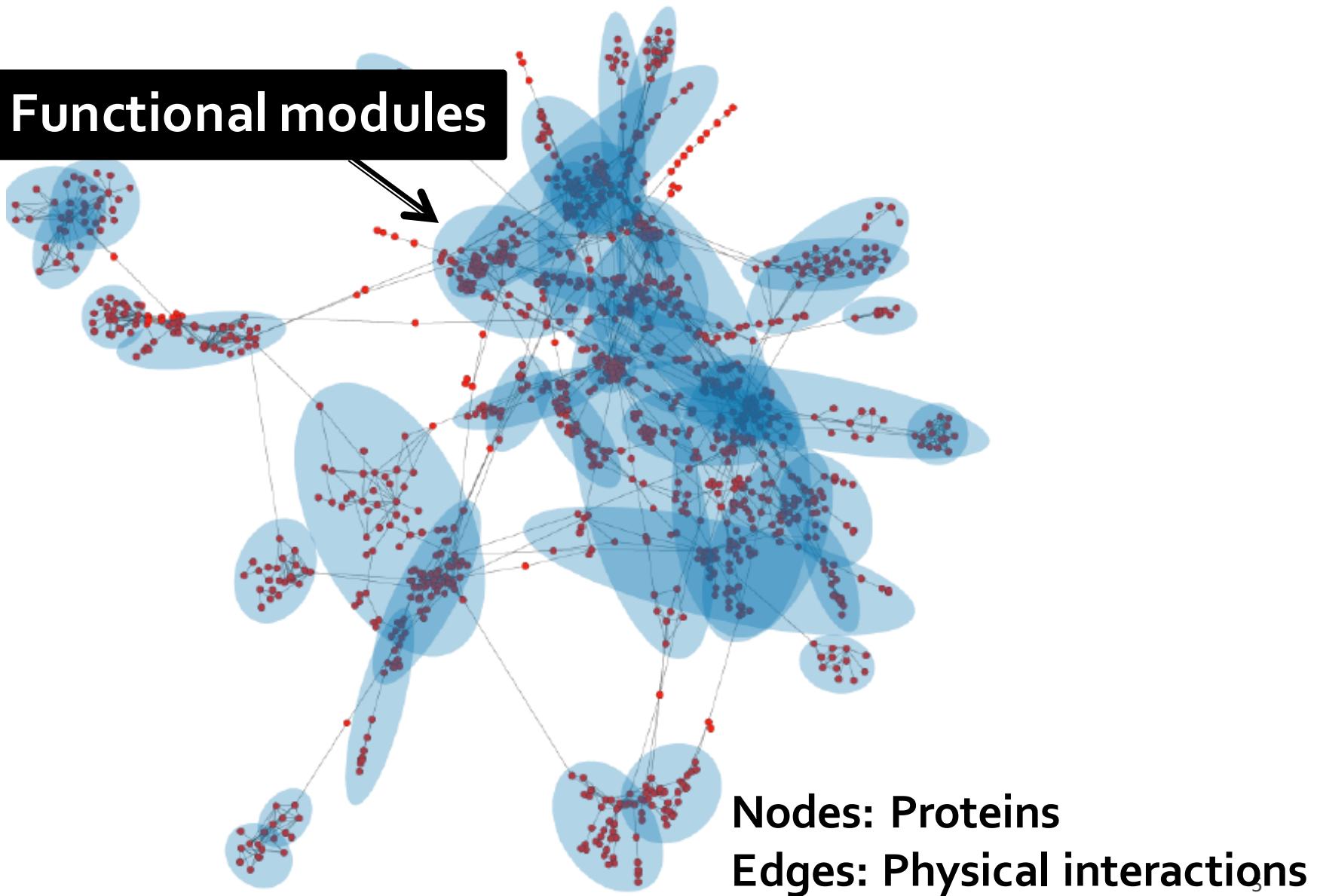
Nodes: Football Teams
Edges: Games played

Protein-Protein Interactions

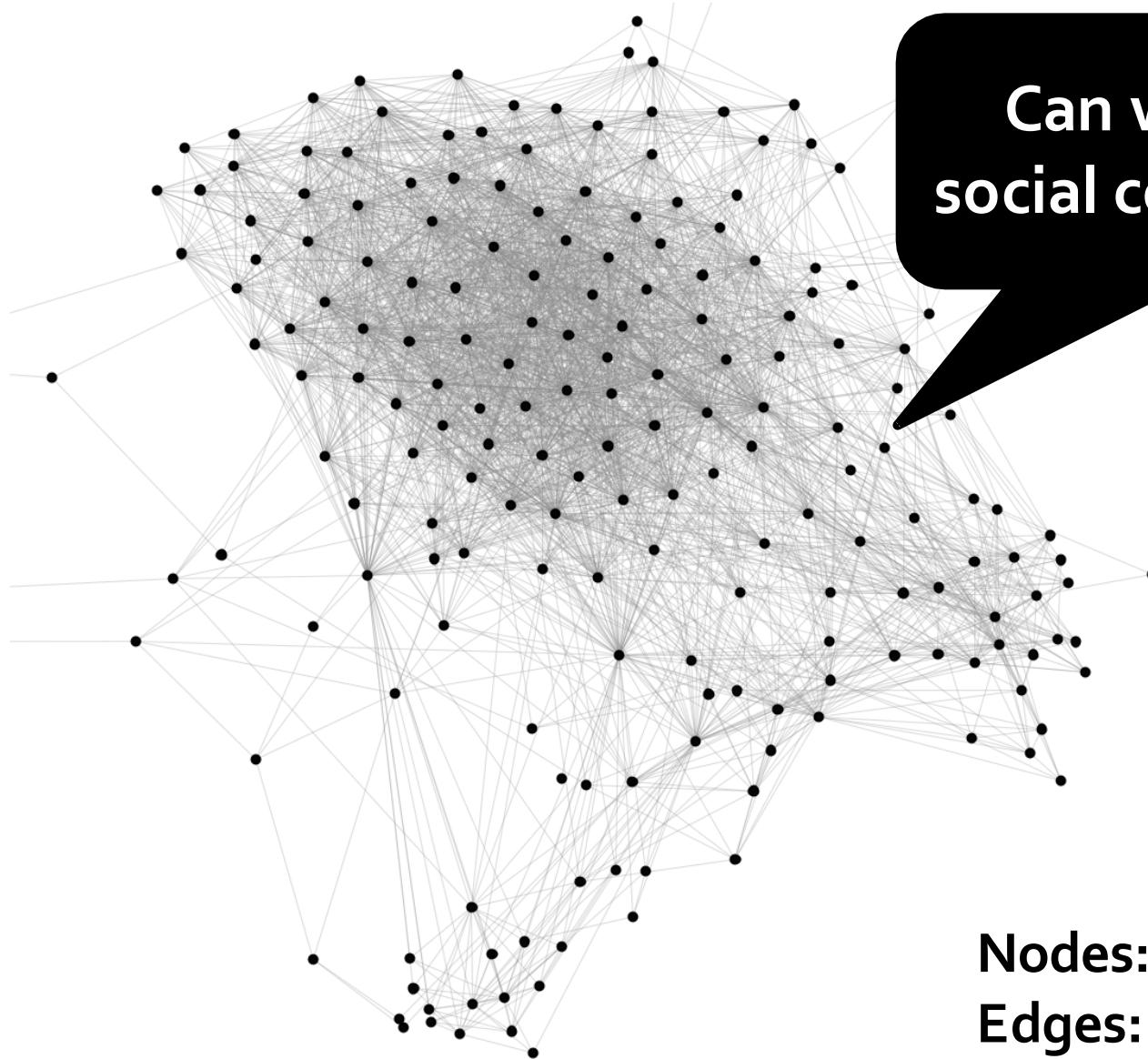


Can we identify
functional modules?

Protein-Protein Interactions



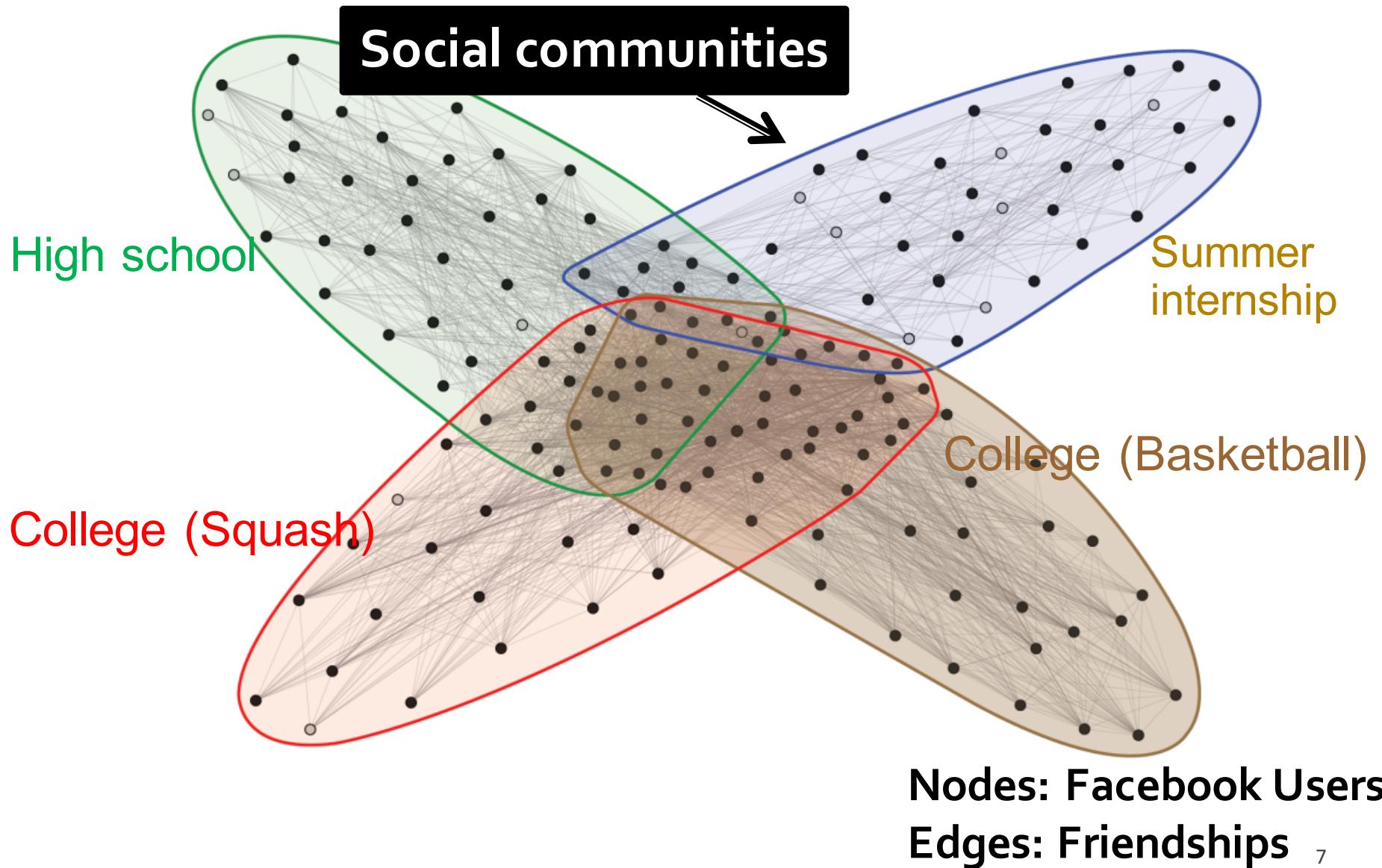
Facebook Network



Can we identify
social communities?

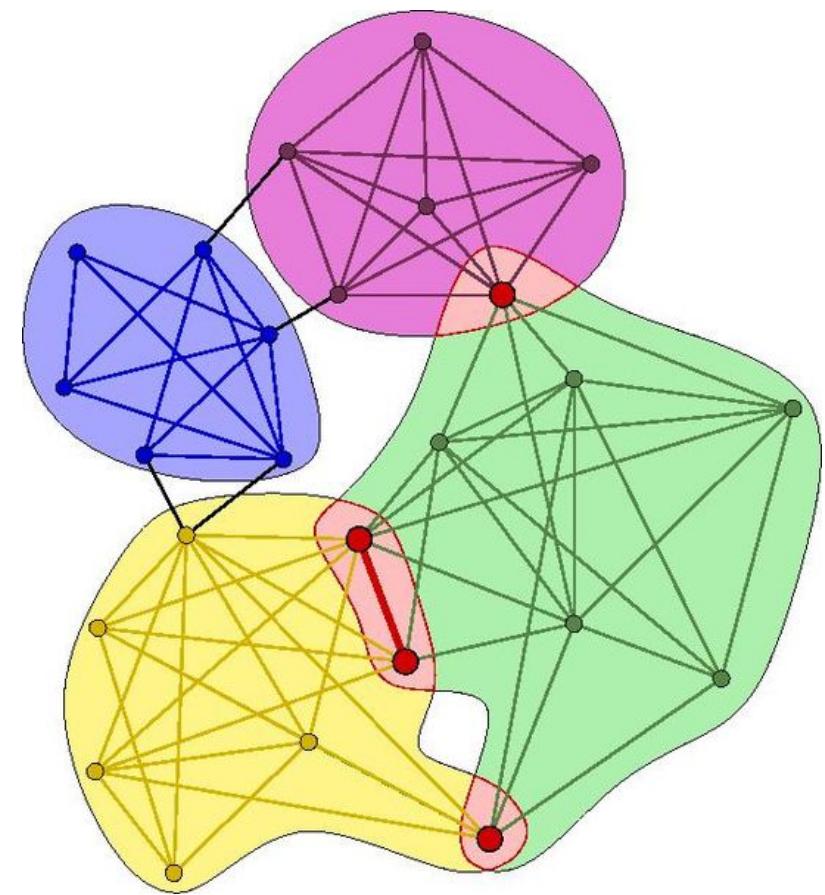
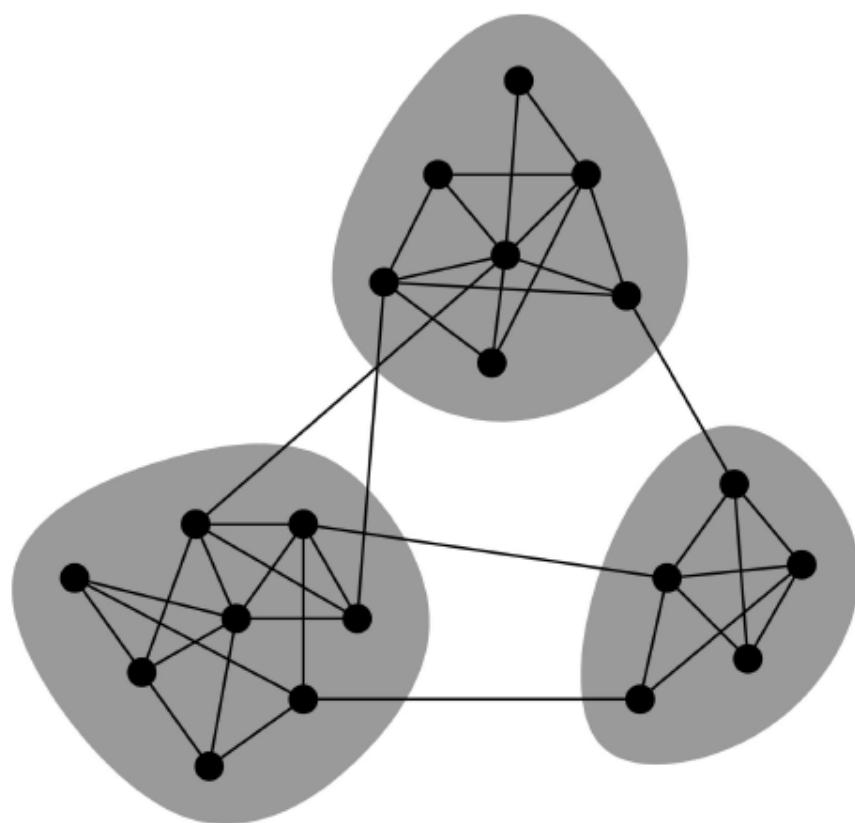
Nodes: Facebook Users
Edges: Friendships

Facebook Network

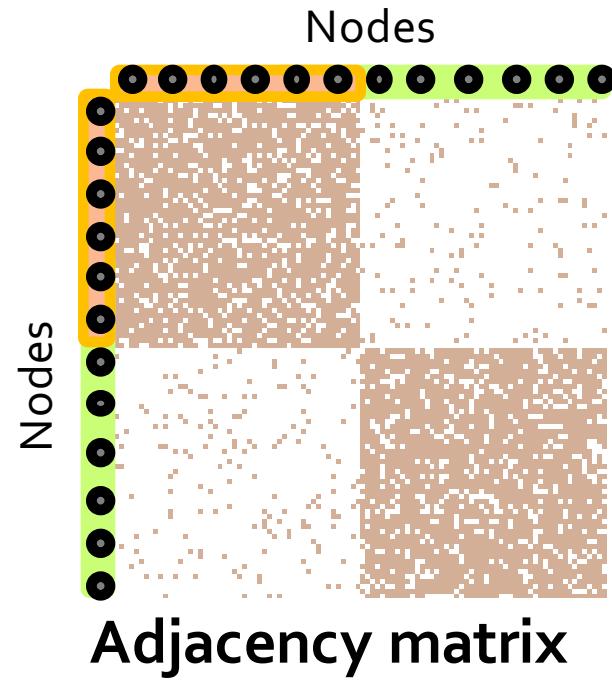
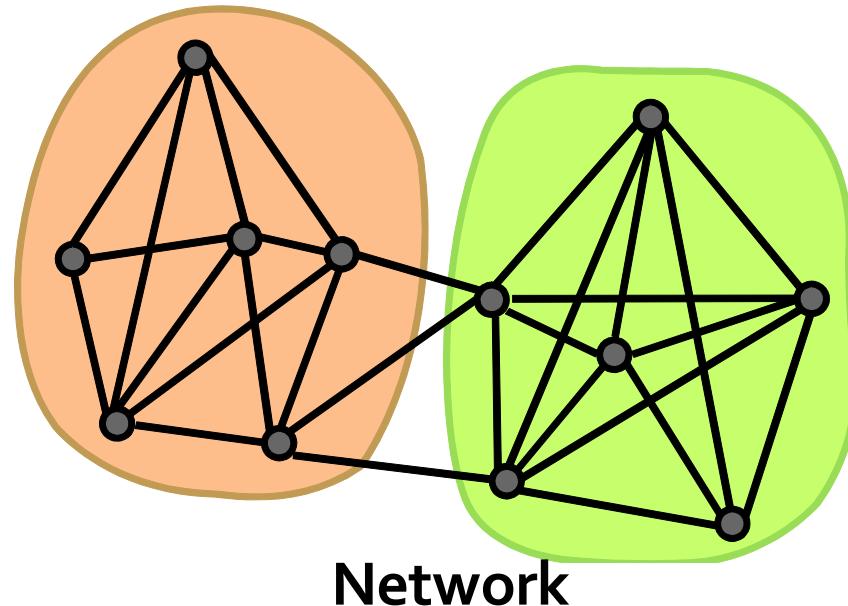


Overlapping Communities

- Non-overlapping vs. overlapping communities

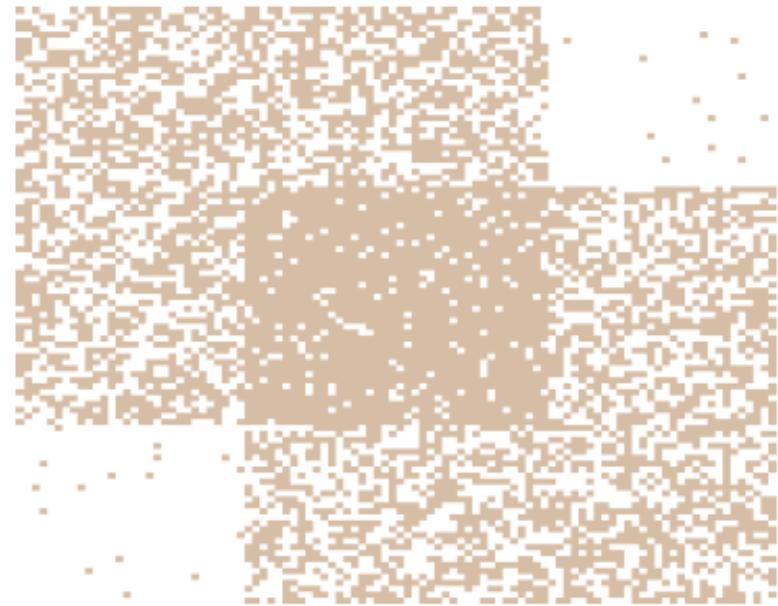
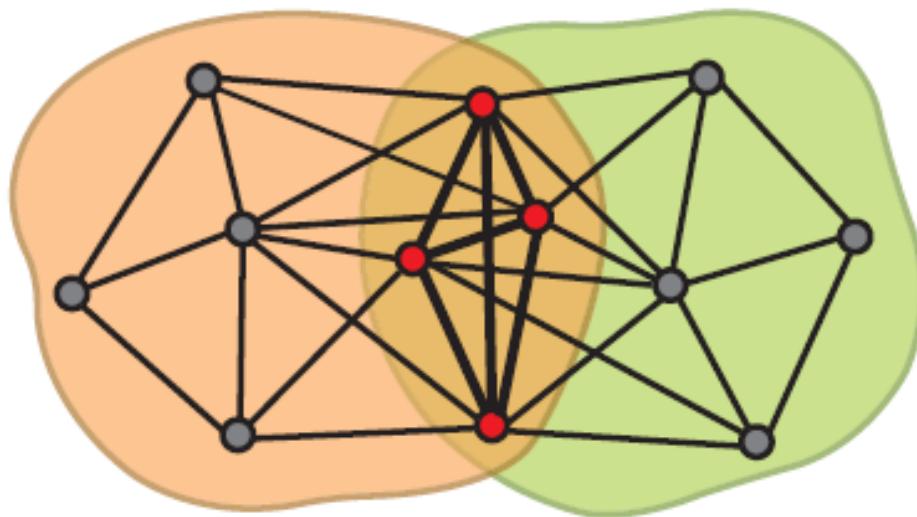


Non-overlapping Communities



Communities as Tiles!

- What is the structure of community overlaps:
Edge density in the overlaps is higher!

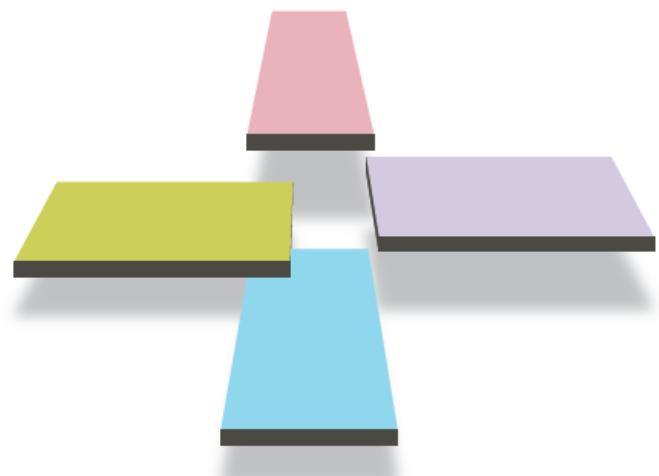


Communities as “tiles”

Recap so far...



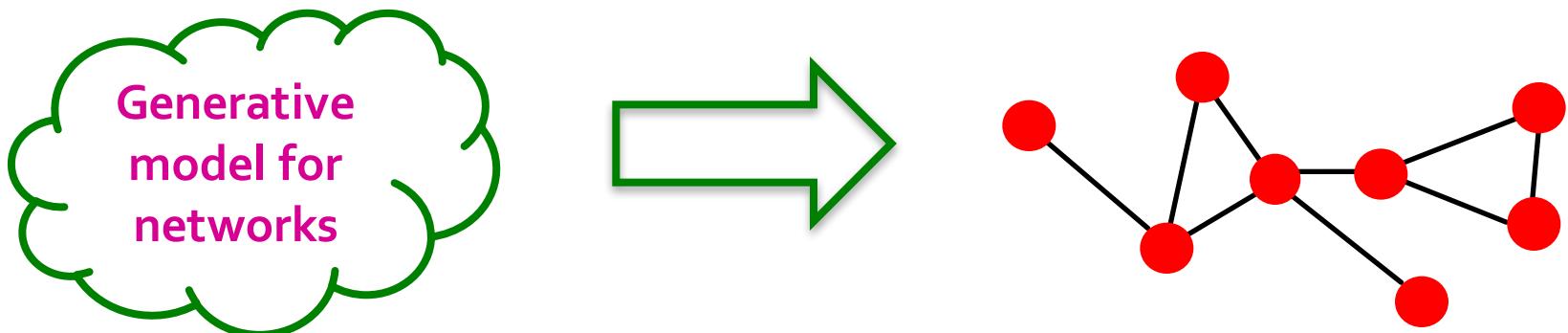
Communities
in a network



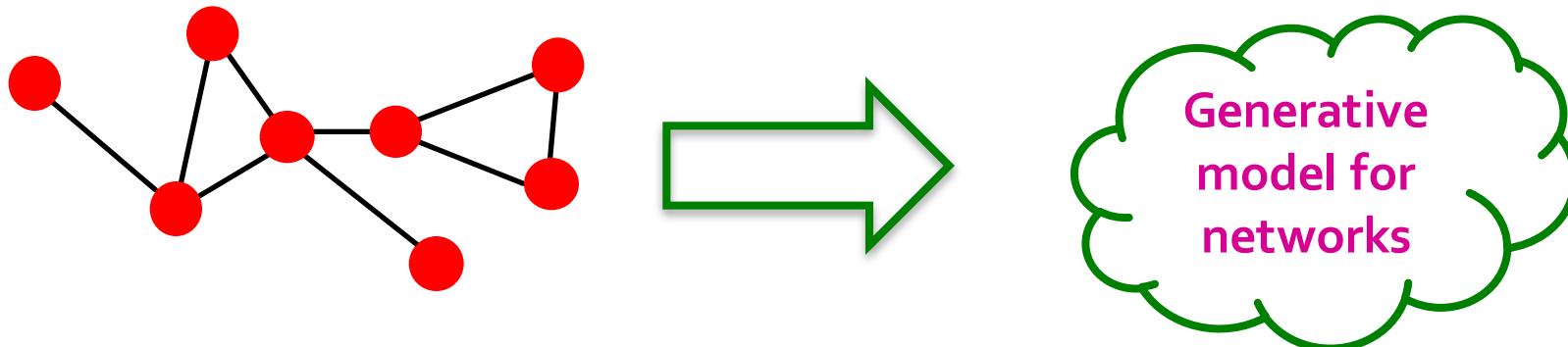
This is what we want!

Plan of attack

- 1) Given a model, we generate the network:

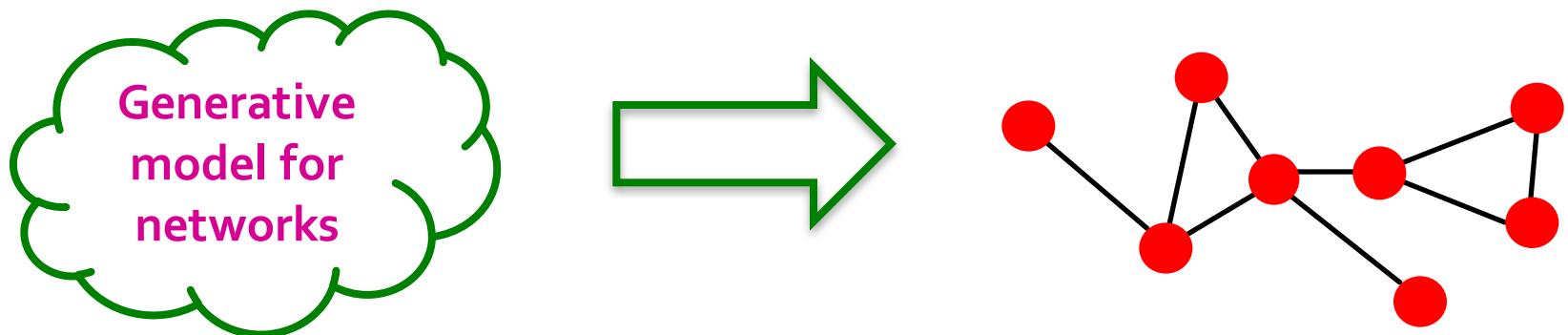


- 2) Given a network, find the “best” model



Model of networks

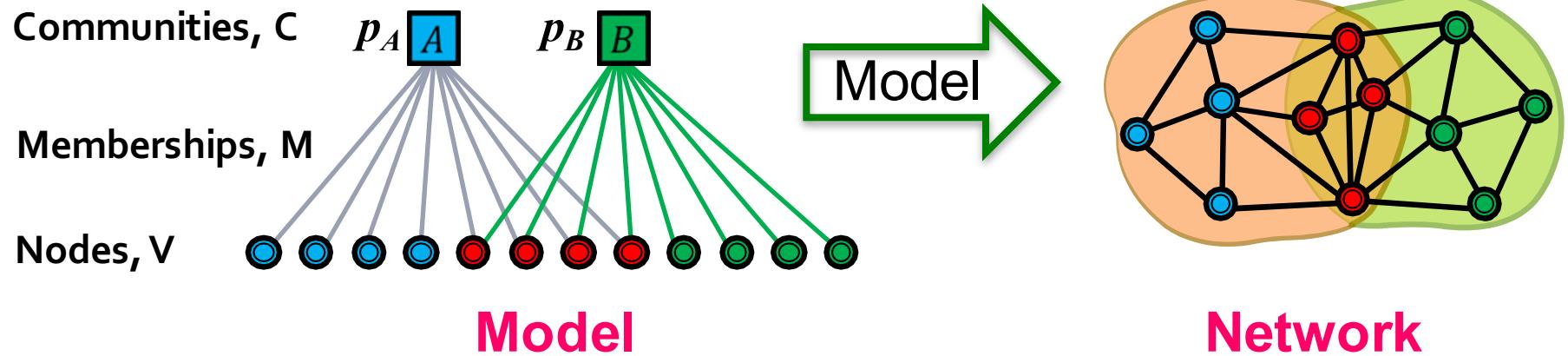
- Goal: Define a model that can generate networks
 - The model will have a set of “parameters” that we will later want to estimate (and detect communities)



- Q: Given a set of nodes, how do communities “generate” edges of the network?

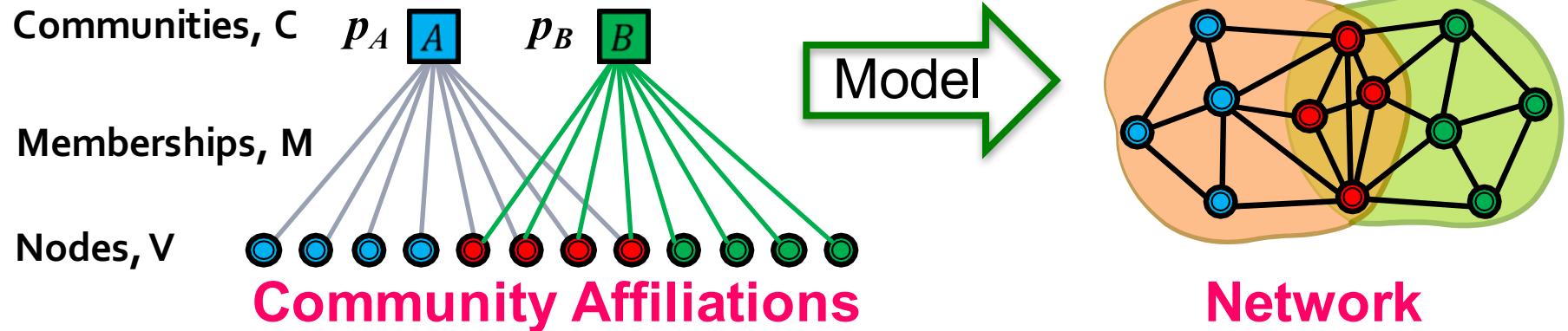
Affiliation-Graph Model

Community-Affiliation Graph



- **Generative model $B(V, C, M, \{p_c\})$ for graphs:**
 - Nodes V , Communities C , Memberships M
 - Each community c has a single probability p_c
 - Later we fit the model to networks to detect communities.

AGM: Generative Process (2)



- **Affiliation Graph Model (AGM) generates the links:**
For each pair of nodes in community A , we connect them with prob. p_A

- **The overall edge probability is:**

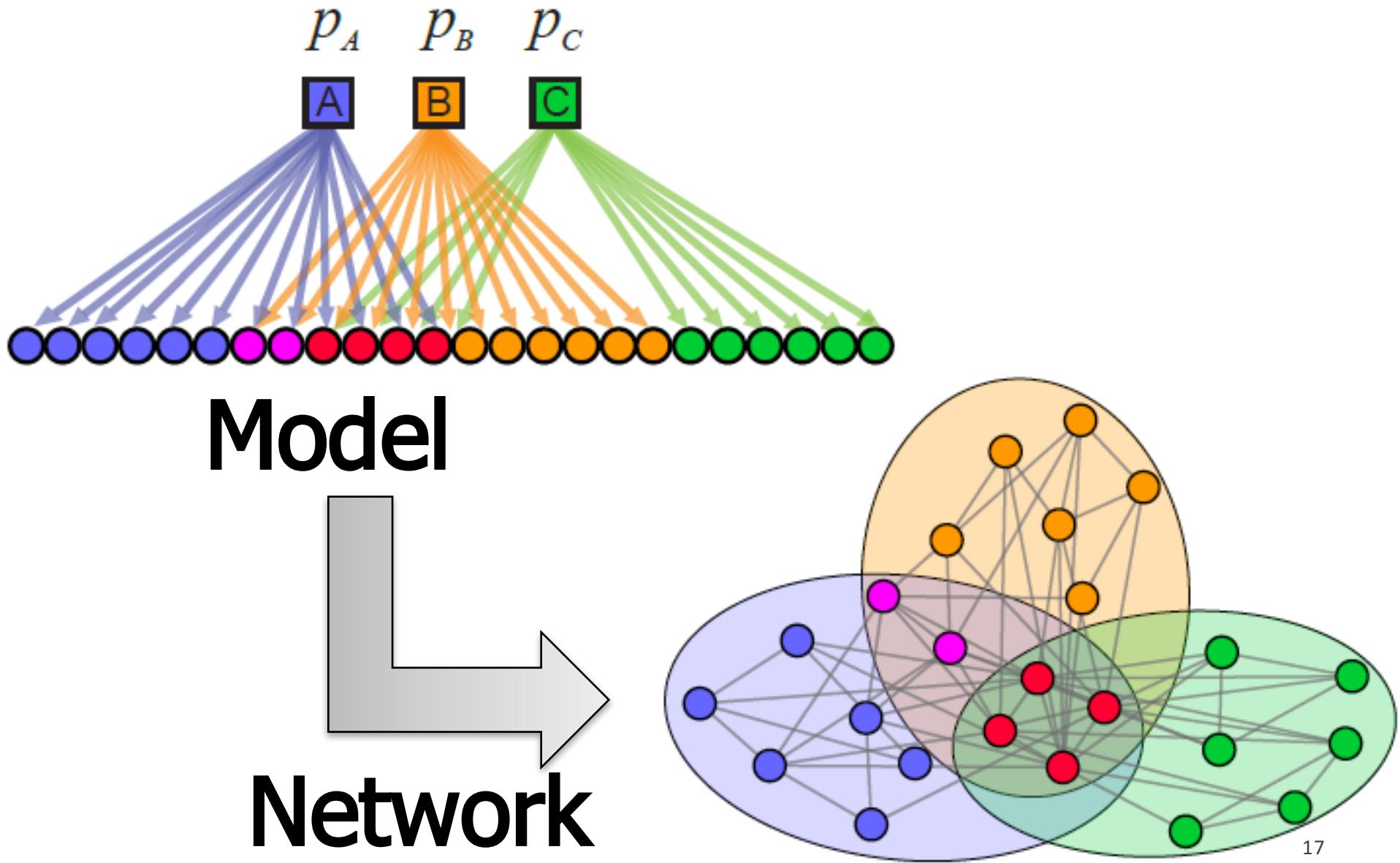
$$P(u, v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

M_u ... set of communities node u belongs to

If u, v share no communities: $P(u, v) = \epsilon$ *epsilon ϵ is a very tiny number*

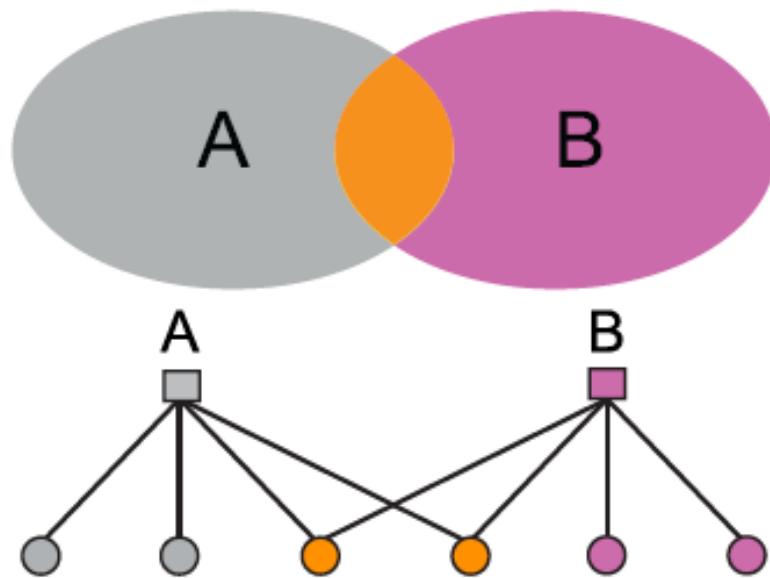
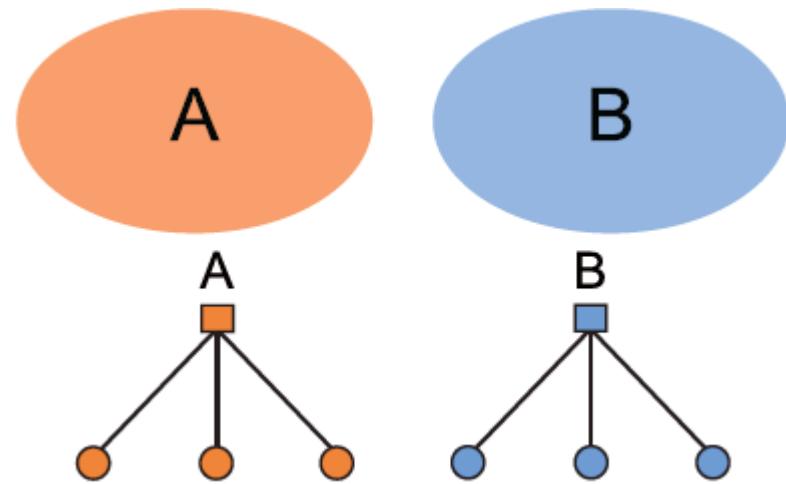
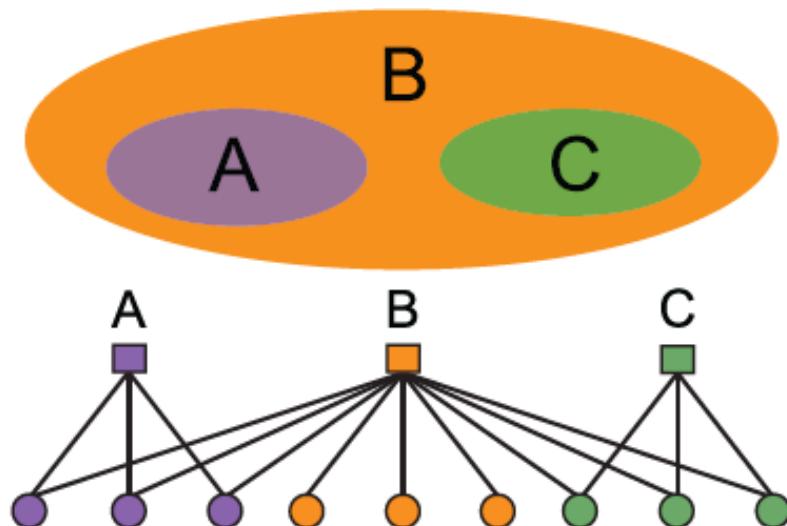
Think of this as an “OR” function: If at least 1 community says “YES” we create an edge

Recap: AGM networks



AGM: Flexibility

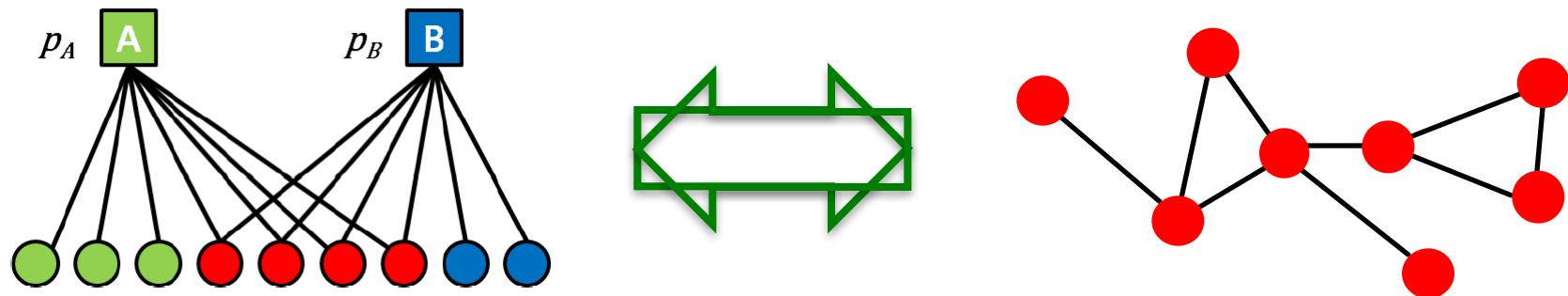
- AGM can express a variety of community structures:
Non-overlapping,
Overlapping, Nested



How do we detect communities with AGM?

Detecting Communities

■ Detecting communities with AGM:



Given a Graph $G(V, E)$, find the Model

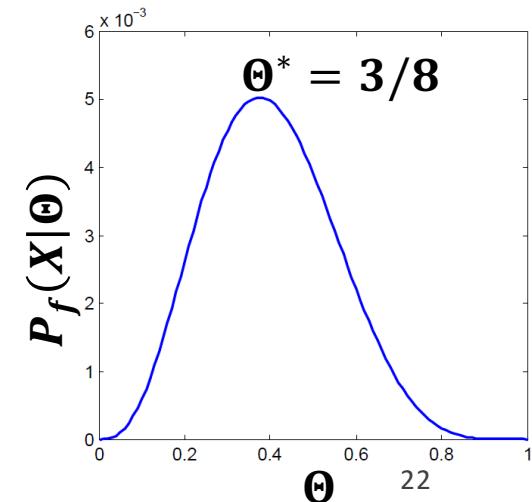
- 1) Affiliation graph M
- 2) Number of communities C
- 3) Parameters p_c

Maximum Likelihood Estimation

- **Maximum Likelihood Principle (MLE):**
 - **Given:** Data X
 - **Assumption:** Data is generated by some model $f(\Theta)$
 - f ... model
 - Θ ... model parameters
 - Want to estimate $P_f(X|\Theta)$:
 - The probability that our model f (with parameters Θ) generated the data
 - **Now let's find the most likely model (including the parameters) that could have generated the data:**
$$\arg \max_{\Theta} P_f(X|\Theta).$$

Example: MLE

- Imagine we are given a set of coin flips
- Task: Figure out the bias of a coin!
 - Data: Sequence of coin flips: $X = [1, 0, 0, 0, 1, 0, 0, 1]$
 - Model: $f(\Theta)$ = return 1 with prob. Θ ,
else return 0 with prob. $1 - \Theta$
 - What is $P_f(X|\Theta)$? Assuming coin flips are independent
 - So, $P_f(X|\Theta) = P_f(1|\Theta) * P_f(0|\Theta) * P_f(0|\Theta) ... * P_f(1|\Theta)$
 - What is $P_f(1|\Theta)$? Simple, $P_f(1|\Theta) = \Theta$
 - Then, $P_f(X|\Theta) = \Theta^3(1 - \Theta)^5$
 - For example:
 - $P_f(X|\Theta = 0.5) = 0.003906$
 - $P_f(X|\Theta = \frac{3}{8}) = 0.005029$
 - What did we learn? Our data was most likely generated by coin with bias $\Theta = 3/8$



AGM Example (1)

- Using AGM for modeling the graph below
- Assumptions (preset some of the parameters):
 - There are two communities C and D with P_C and P_D
 - We have determined (or are using as a temporary hypothesis) the community memberships:
 - $C = \{w, x, y\}$ and $D = \{w, y, z\}$

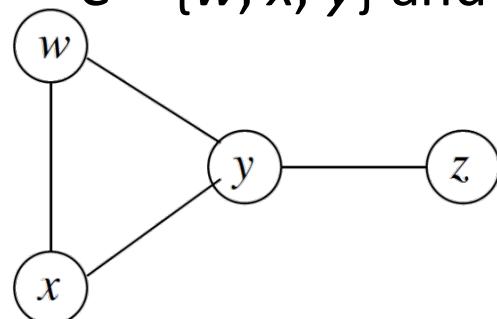


Figure 10.20: A social graph

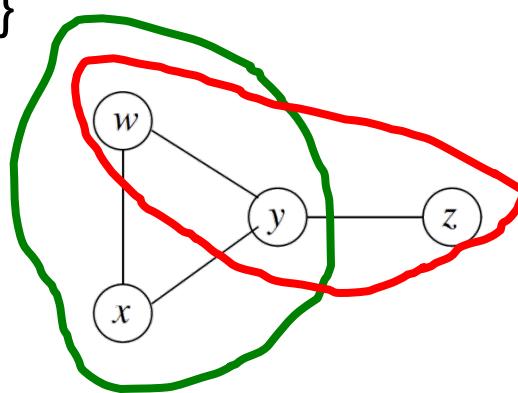
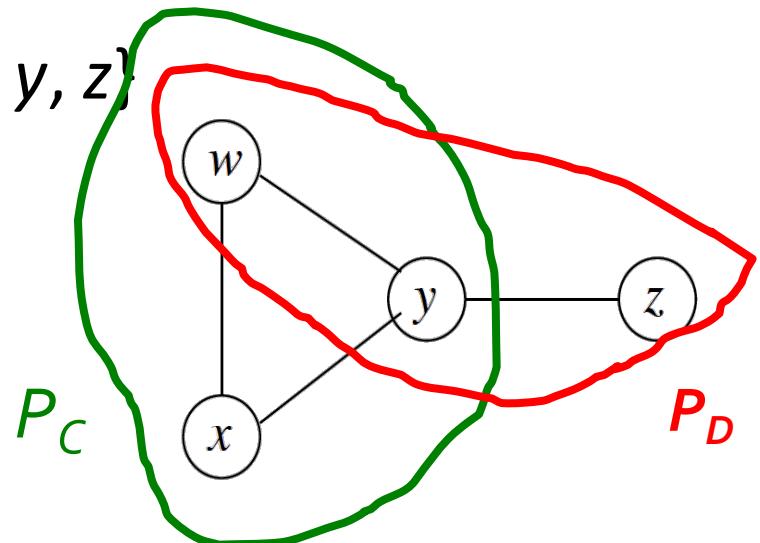


Figure 10.20: A social graph

AGM Example (cont.)

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

- Given $C = \{w, x, y\}$ and $D = \{w, y, z\}$
- For w an x , $M_{wx} = \{C\}$
 - $P_{wx} = 1 - (1 - P_C) = P_C$
- And
 - $P_{xy} = P_C$ and $P_{yz} = P_{wz} = P_D$
 - $P_{wy} = 1 - (1 - P_C)(1 - P_D)$
 - $P_{xz} = \text{a tiny value}$
- The likelihood of this graph given the membership assumption
- $P_{wx} P_{wy} P_{xy} P_{yz} (1 - P_{wz})(1 - P_{xz})$
 - $(p_C)^2 p_D (p_C + p_D - p_C p_D)(1 - p_D)(1 - \epsilon)$

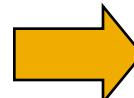


Maximum-Likelihood Estimation for Graphs

- How do we do MLE for graphs?
 - Model generates a **probabilistic adjacency matrix**
 - We then flip all the entries of the probabilistic matrix to obtain the **binary adjacency matrix A**

For every pair
of nodes u, v
AGM gives the
prob. p_{uv} of
them being
linked

0	0.10	0.10	0.04
0.10	0	0.02	0.06
0.10	0.02	0	0.06
0.04	0.06	0.06	0

Flip
biased
coins


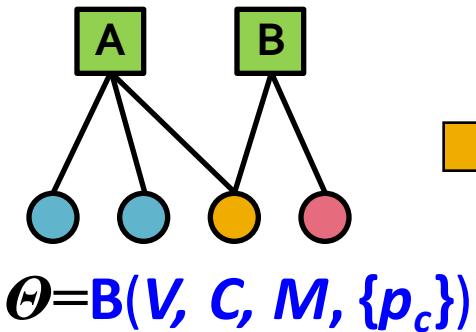
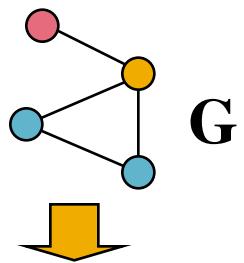
A	0	1	0	0
0	1	0	1	1
1	0	1	0	1
0	1	0	1	0
0	1	1	0	0

- The likelihood of AGM generating graph G:

$$P(G | \Theta) = \prod_{(u,v) \in E} P(u,v) \prod_{(u,v) \notin E} (1 - P(u,v))$$

Graphs: Likelihood $P(G|\Theta)$

- Given graph $G(V,E)$ and Θ , we calculate likelihood that Θ generated G : $P(G|\Theta)$



0	0.9	0.9	0
0.9	0	0.9	0
0.9	0.9	0	0.9
0	0	0.9	0

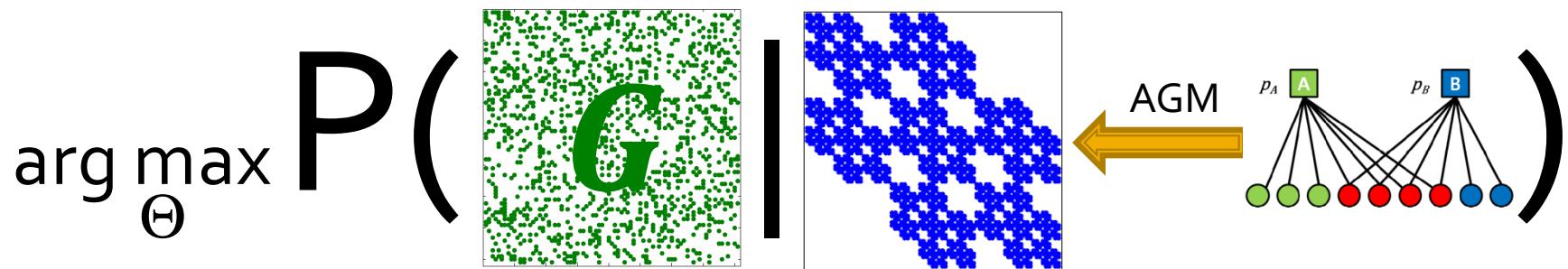
0	1	1	0
1	0	1	0
1	1	0	1
0	0	1	0

$P(G|\Theta)$

$$P(G | \Theta) = \prod_{(u,v) \in E} P(u, v) \prod_{(u,v) \notin E} (1 - P(u, v))$$

MLE for Graphs

- Our goal: Find $\Theta = B(V, C, M, \{p_C\})$ such that:



- How do we find $B(V, C, M, \{p_C\})$ that maximizes the likelihood?

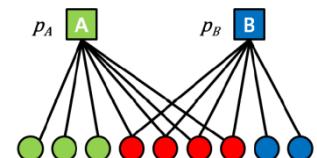
MLE for AGM

- Our goal is to find $B(V, C, M, \{p_C\})$ such that:

$$\arg \max_{B(V, C, M, \{p_C\})} \prod_{u, v \in E} P(u, v) \prod_{uv \notin E} (1 - P(u, v))$$

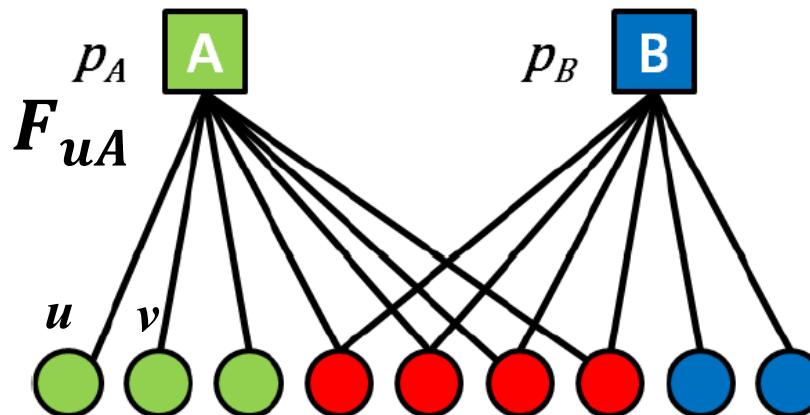
- Problem: Finding \mathbf{B} means finding the bipartite affiliation network.

- There is no nice way to do this.
- Fitting $B(V, C, M, \{p_C\})$ is too hard, let's change the model (so it is easier to fit)!



From AGM to BigCLAM (1)

- **Relaxation:** Memberships have strengths (avoid discrete membership changes)

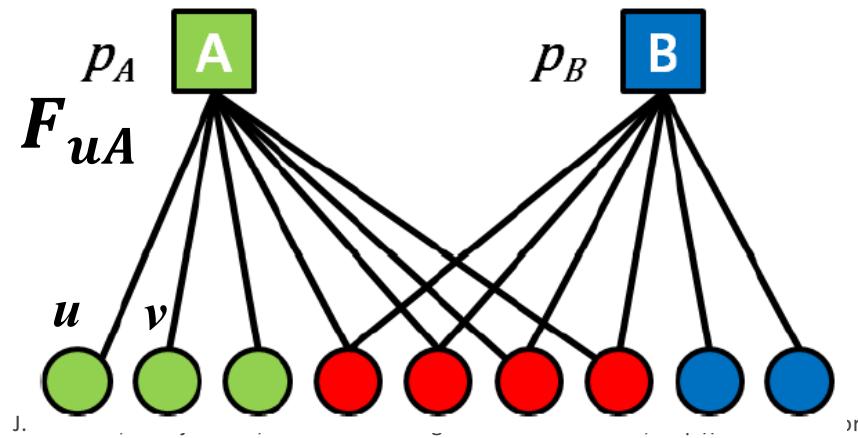


- F_{uA} : The membership strength of node u to community A (> 0) ($F_{uA} = 0$: no membership)
- Each community A links nodes independently.
Define:

$$P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$$

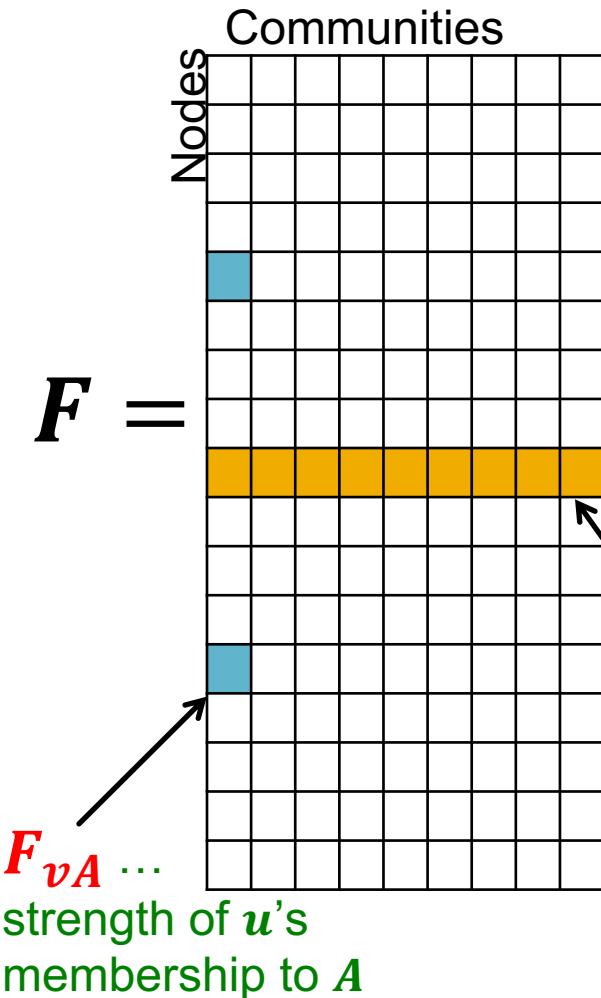
From AGM to BigCLAM (2)

- $P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$
- If u and v are not in the same community
 - $P_A(u, v) = 1 - \exp(0) = 0$
- If F_{uA} and F_{vA} are both large numbers:
 - $P_A(u, v) = 1 - e^{-ALargeNumber} = 1 - ASmallNumber$
- If F_{uA} and F_{vA} are both small numbers:
 - $P_A(u, v) = 1 - e^{-ASmallNumber} = 1 - ALargeNumber.$



Factor Matrix F

■ Community membership strength matrix F



- $P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$
 - Probability of connection is proportional to the product of strengths
 - Notice: If one node doesn't belong to the community ($F_{uC} = 0$) then $P(u, v) = 0$
- Prob. that **at least one** common community C links the nodes:
 - $P(u, v) = 1 - \prod_C (1 - P_C(u, v))$

From AGM to BigCLAM

- Community A links nodes u, v independently:

$$P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$$

- Then prob. at least one common C links them:

$$\begin{aligned} P(u, v) &= 1 - \prod_C (1 - P_C(u, v)) \\ &= 1 - \exp(-\sum_C F_{uC} \cdot F_{vC}) \\ &= 1 - \exp(-F_u \cdot F_v^T) \end{aligned}$$

- Example F matrix:

$$F_u : \begin{array}{|c|c|c|c|} \hline 0 & 1.2 & 0 & 0.2 \\ \hline \end{array}$$

$$F_v : \begin{array}{|c|c|c|c|} \hline 0.5 & 0 & 0 & 0.8 \\ \hline \end{array}$$

$$F_w : \begin{array}{|c|c|c|c|} \hline 0 & 1.8 & 1 & 0 \\ \hline \end{array}$$

Node community
membership strengths

Then: $F_u \cdot F_v^T = 0.16$

And: $P(u, v) = 1 - \exp(-0.16) = 0.14$

But: $P(u, w) = 0.88$

$P(v, w) = 0$

BigCLAM: How to find F

- **Task:** Given a network $G(V, E)$, estimate F
 - Find F that maximizes the likelihood:

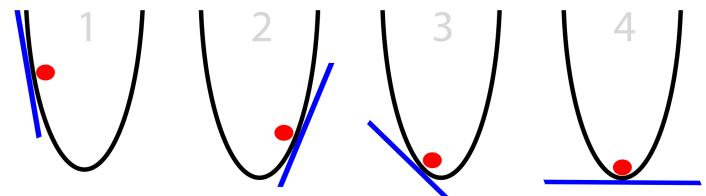
$$\arg \max_F \prod_{(u,v) \in E} P(u, v) \prod_{(u,v) \notin E} (1 - P(u, v))$$

- where: $P(u, v) = 1 - \exp(-F_u \cdot F_v^T)$
- Many times we take the logarithm of the likelihood, and call it log-likelihood: $l(F) = \log P(G|F)$

- **Goal:** Find F that maximizes $l(F)$:

$$l(F) = \sum_{(u,v) \in E} \log(1 - \exp(-F_u F_v^T)) - \sum_{(u,v) \notin E} F_u F_v^T$$

BigCLAM: V1.0



$$l(F_u) = \sum_{v \in \mathcal{N}(u)} \log(1 - \exp(-F_u F_v^T)) - \sum_{v \notin \mathcal{N}(u)} F_u F_v^T$$

- Compute gradient of a single row F_u of F :

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

- Coordinate gradient ascent:
 - Iterate over the rows of F :
 - Compute gradient $\nabla l(F_u)$ of row u (while keeping others fixed)
 - Update the row F_u : $F_u \leftarrow F_u + \eta \nabla l(F_u)$
 - Project F_u back to a non-negative vector: If $F_{uC} < 0$: $F_{uC} = 0$
 - This is slow! Computing $\nabla l(F_u)$ takes linear time!

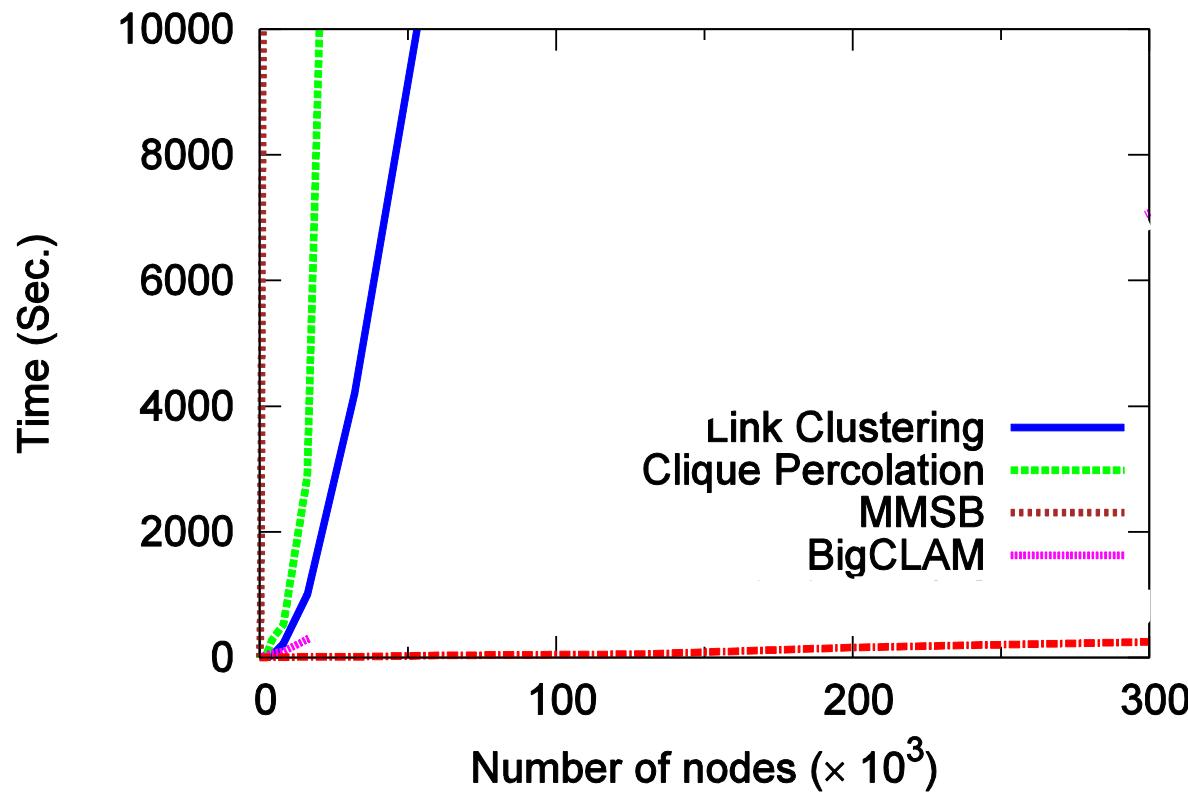
BigCLAM: V2.0

- However, we notice:

$$\sum_{v \notin \mathcal{N}(u)} F_v = \left(\sum_v F_v - F_u - \sum_{v \in \mathcal{N}(u)} F_v \right)$$

- We cache $\sum_v F_v$ **<-- Compute in offline**
- So, computing $\sum_{v \in \mathcal{N}(u)} F_v$ now takes **linear time** in the degree $|\mathcal{N}(u)|$ of u
 - In networks degree of a node is much smaller to the total number of nodes in the network, so this is a significant speedup!

BigClam: Scalability



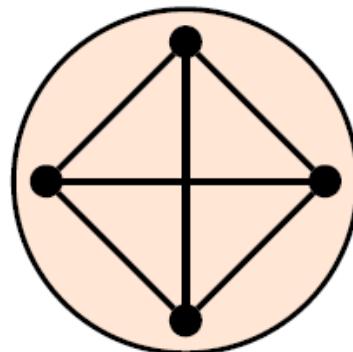
- **BigCLAM takes 5 minutes for 300k node nets**
 - Other methods take 10 days
- **Can process networks with 100M edges!**

Extension: Directed memberships

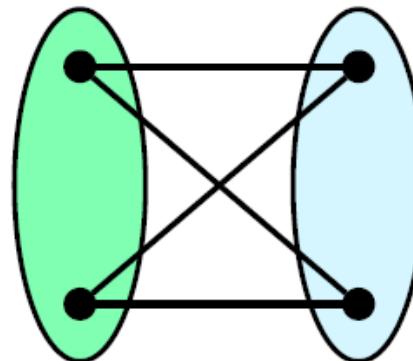
Extension: Beyond Clusters

Undirected

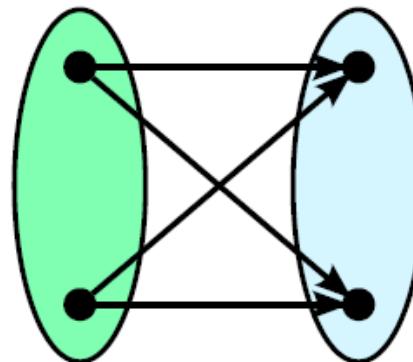
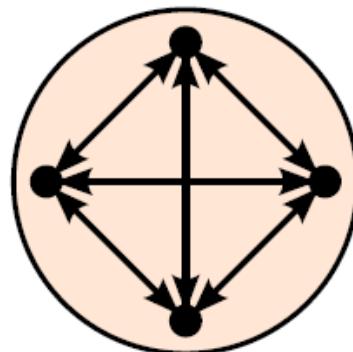
Cohesive



2-mode

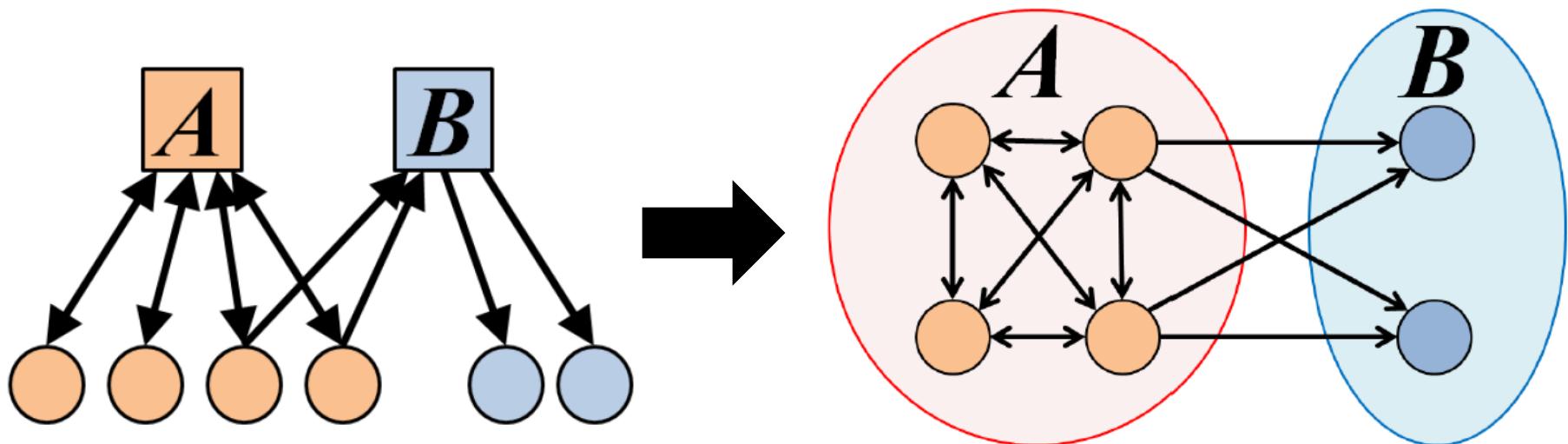


Directed

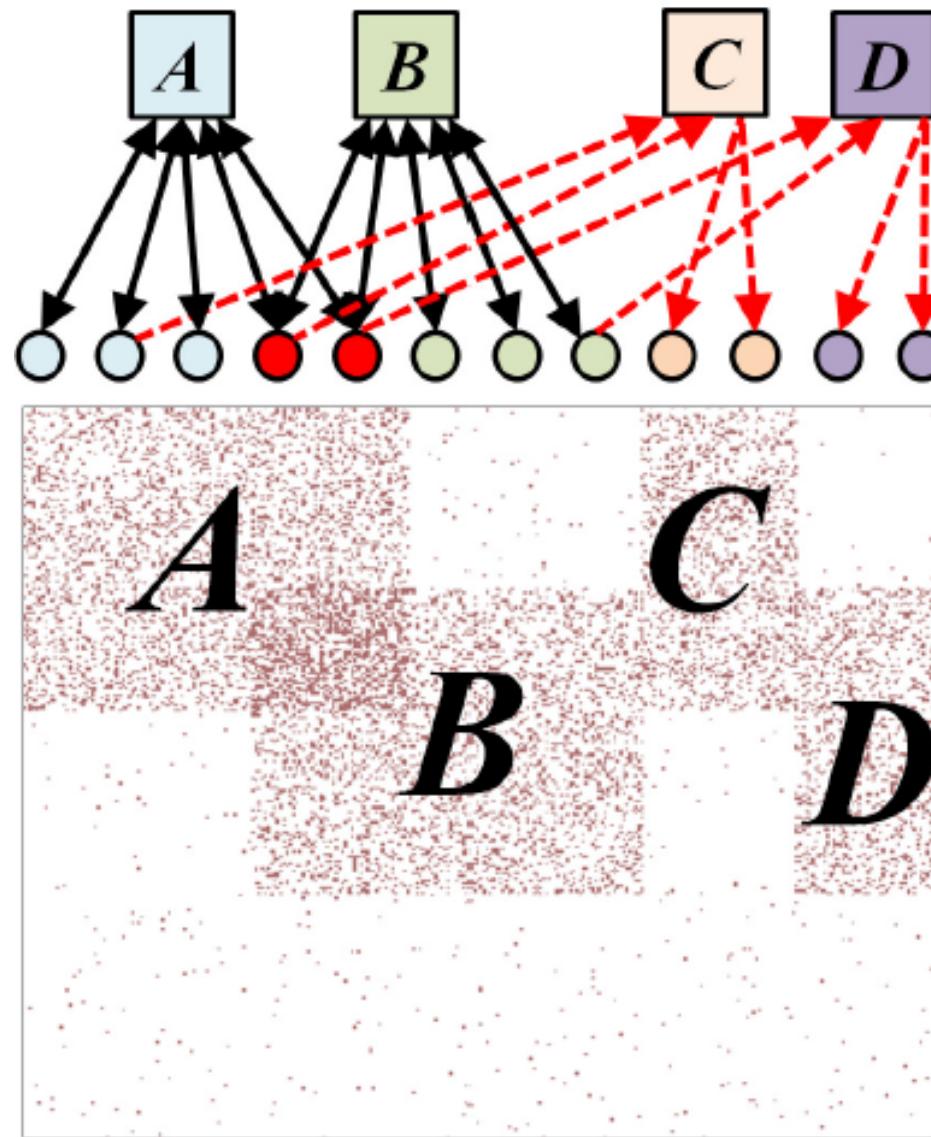


Extension: Directed AGM

- **Extension:**
Make community membership edges directed!
 - Outgoing membership: Nodes “**sends**” edges
 - Incoming membership: Node “**receives**” edges



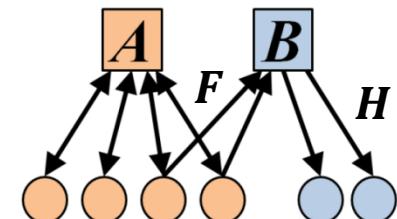
Example: Model and Network



Directed AGM

- Everything is almost the same except now we have 2 matrices: F and H

- F ... out-going community memberships
- H ... in-coming community memberships

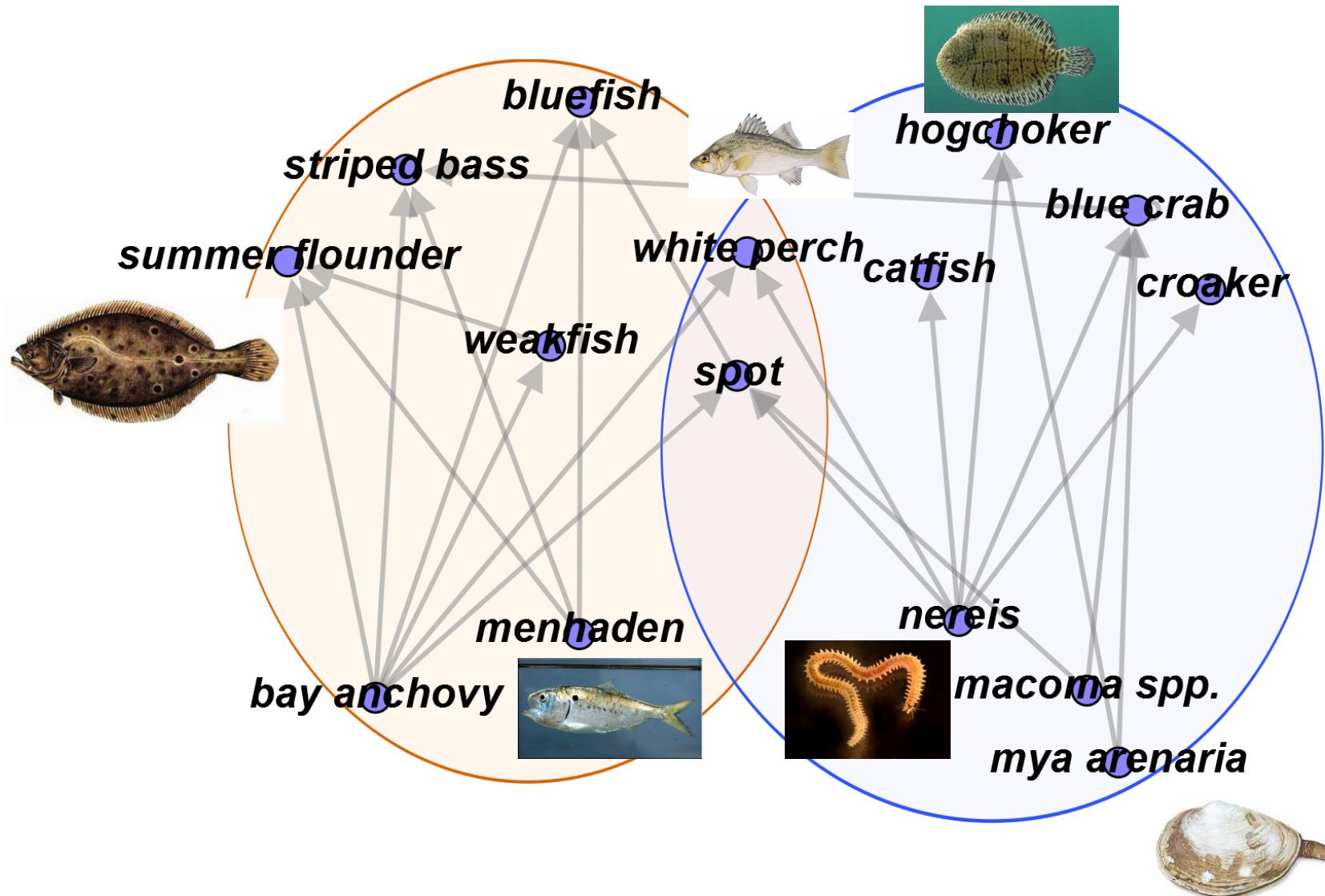


- Edge prob.: $P(u, v) = 1 - \exp(-F_u H_v^T)$
- Network log-likelihood:

$$l(F, H) = \sum_{(u, v) \in E} \log(1 - \exp(-F_u H_v^T)) - \sum_{(u, v) \notin E} F_u H_v^T$$

which we optimize the same way as before

Predator-prey Communities



More details at...

- [Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach](#) by J. Yang, J. Leskovec. *ACM International Conference on Web Search and Data Mining (WSDM)*, 2013.
- [Detecting Cohesive and 2-mode Communities in Directed and Undirected Networks](#) by J. Yang, J. McAuley, J. Leskovec. *ACM International Conference on Web Search and Data Mining (WSDM)*, 2014.
- [Community Detection in Networks with Node Attributes](#) by J. Yang, J. McAuley, J. Leskovec. *IEEE International Conference On Data Mining (ICDM)*, 2013.

Mini Project-Deadline Nov 15, 2019

NLP for Social Media (2nd ed.)

Synthesis Lectures on
Human Language Technologies

Authors : A. Farzindar and D. Inkpen

Analysis of studies including more
than 400 references.

-Mini project: read two papers from the book
and write a summary statement (2 pages)

-Write your review on Amazon

