Social Networks

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
July 8, 2021

Learning Goals Today / Overview

- Intro: Social Networks are Graphs
- How to Cluster Social Networks into Groups
- Non-overlapping communities: the Girvan-Newman Algorithm
- Overlapping communities: the Graph Affiliation Model
- Direct Discovery of Overlapping Communities

Social Networks as Graphs

Social Networks: Introduction

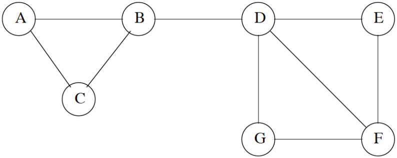
Basic Examples

- Facebook, Twitter, Google+

Defining Properties

- Collection of entities participating in network
  - Usually people, but other entities conceivable
- There is a relationship between the entities
  - Being friends is frequent relationship
  - Relationship can be of 0-1 type, or weighted
- Assumption of nonrandomness or locality
  - Hard to formalize, intuition is that relationships tend to cluster
  - If entity A is related with both B and C, B and C are related with larger probability

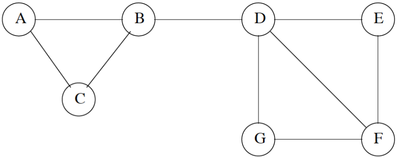
Social Network Graphs: Entities and Relationships



Adopted from `mmds.org`

- *Entities:* Nodes A to G
- *Relationships:* Represented by edges between nodes
  - *Example:* A is “friends” with B and C

Social Network Graphs: Locality



Adopted from `mmds.org`

- *Locality:*
  - There are 9 out of 21 possible edges: $\frac{9}{21} = 0.429$
  - Given nodes X, Y, Z such that there are edges $(X, Y), (Y, Z)$, random occurrence of (X, Z) is $\frac{7}{19} = 0.368$
  - However, across all pairs of existing edges $(X, Y), (Y, Z)$, probability that (X, Z) exists is $\frac{9}{16} = 0.563$
  - ☞ Network exhibits locality

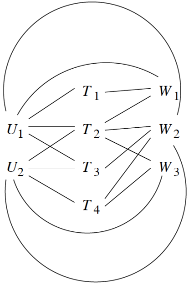
Social Networks: Examples

- *Telephone Networks:*
  - *Nodes* are phone numbers, *edges* exist if one number called another
  - *Edge weights:* Number of calls (within certain period of time)
  - *Communities:* Groups of friends, members of a club, people working at same company
- *Email Networks:*
  - *Nodes* are email addresses, *edges* indicate exchange of emails
  - *Edge directionality* may matter, so graph with directed edges
  - *Communities:* Similar to telephone networks

Social Networks: Examples

- *Collaboration Networks:*
  - *Nodes* e.g. represent authors, *edges* indicate working on same document
  - *Alternatively:* nodes represent documents, edges indicate that identical author contributed
  - *Communities:* Groups interested in / working on same subjects; documents sharing related content
- *Other:*
  - *Information networks:* Documents, web graphs, patents
  - *Infrastructure networks:* Roads, planes, water pipes, power grids
  - *Biological networks:* Genes, proteins, drugs
  - *Product co-purchasing networks:* E.g. Groupon

Several Types of Nodes



Adopted from `mmds.org`

Examples

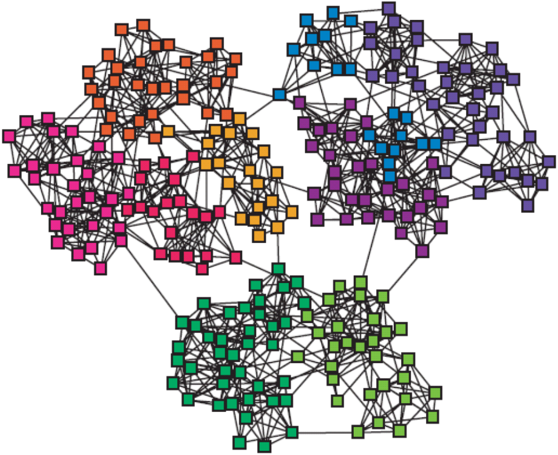
- Figure: Users (U) put tags (T) on web pages (W): tri-partite network
- Put documents and authors into one bi-partite network

Clustering Social Networks

Clustering Social Networks: Introduction

- An important aspect of social networks are *communities*
- Communities reveal themselves as groups of nodes that share unusually many edges
- Clustering social networks relates to the discovery of such communities

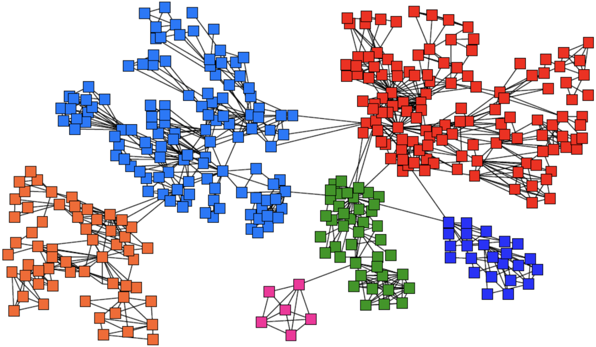
Communities



Differently Colored Communities in Social Network

Adopted from `mmds.org`

Clustered Network



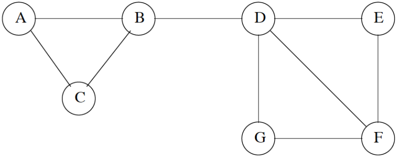
Differently Colored Clusters in Social Network

Adopted from `mmds.org`

Distance Measures in Social Networks

- Standard clustering techniques work with distance measures
- Distance measures are not obvious to define in social networks
  - Let $x, y \in V$ be two nodes in a social network $G = (V, E)$. The measure
    $$d(x,y) = \begin{cases} 0 & (x,y) \in E \\ 1 & (x,y) \notin E \end{cases}$$
    violates the triangle inequality, hence is no distance measure
  - Exchanging 0 with 1, and 1 with ∞ does not help
  - Other binary-valued measures (e.g. 1 and 1.5) agree with triangle inequality
- *But:* Additional issues apply

Social Networks: Clustering Issues



Communities: A-B-C and D-E-F-G

Adopted from `mmds.org`

- *Hierarchical Clustering:* Randomly picks closest nodes/clusters
- Distance between clusters: distance between closest points
- As soon as clusters are joined on B and D, clusters not as desired
- *Summary:* Standard clustering techniques difficult/impossible to sensibly implement

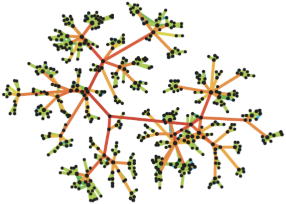
BETWEENNESS

Idea: Identify edges that are least likely to be within community

DEFINITION [BETWEENNESS]
The *betweenness* of an edge (a, b) is

- the number of pairs of nodes (x, y) such that (a, b) makes part of the *shortest path* leading from x to y
- If for (x, y) there are several shortest paths, (a, b) is credited the fraction of shortest paths leading through (a, b) when computing its betweenness

BETWEENNESS



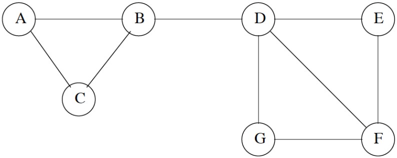
Telephone network:
Links between communities have great betweenness

Adopted from `mmds.org`

Explanation

- High betweenness means that (a, b) is a bottleneck for shortest paths
- If nodes (a, b) lie within community, there are too many options for shortest paths to circumvent (a, b) (so (a, b) gets credited only small fractions)

Betweenness: Example



Adopted from `mmds.org`

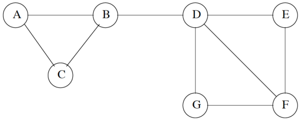
- (B, D) has the greatest betweenness, 12
  - It is on any shortest path between A, B, C and D, E, F, G
- (D, F) has betweenness 4
  - It lies on all shortest paths between A, B, C, D and F

The Girvan-Newman Algorithm

Calculating Betweenness

Algorithmic Principle

- Visit each node X once
- Compute shortest paths from X to any other node Y
- To visit nodes Y from X, perform breadth-first search (BFS)



Social Network; consider BFS from E

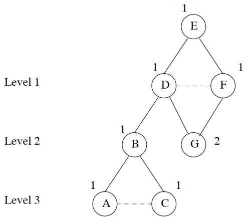
Adopted from `mmds.org`

The Girvan-Newman Algorithm

Calculating Betweenness

Algorithmic Principle

- Visit each node X once
- Compute shortest paths from X to any other node Y
- To visit nodes Y from X, perform breadth-first search (BFS)

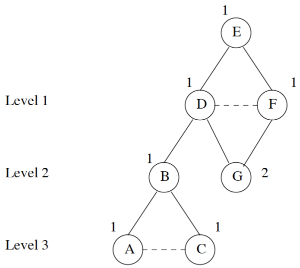


BFS starting from E on social network from slide before

Adopted from `mmds.org`

The Girvan-Newman Algorithm

Calculating Betweenness



BFS starting from E

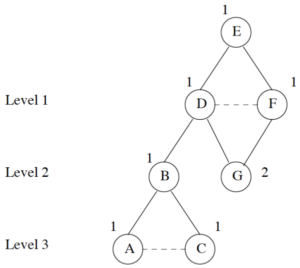
Adopted from `mmds.org`

Intuition / Notation

- Length of shortest path from X to Y: level of BFS starting at X
- Edges within BFS level cannot be part of shortest paths from X
- Edges between different levels are referred to as *DAG (directed acyclic graph) edges*
- DAG edges are on at least one shortest path leaving from X

The Girvan-Newman Algorithm

Calculating Betweenness



BFS starting from E

Adopted from `mmds.org`

Example Notation

- Solid edges = DAG edges: e.g. (D, B), (E, F)
- Dashed edges = within level: e.g. (D, F), (A, C)
- For DAG edge (Y, Z) where Y is closer to root X than Z:
  - Y is said to be the *parent*
  - Z is said to be the *child*

The Girvan-Newman Algorithm

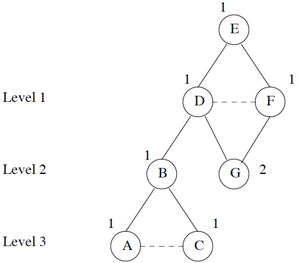
Calculating Betweenness

Two Stages

- *Labeling:* For each node, assign number of shortest paths from root to that node
  - Proceed from root to leaves in BFS order
- *Crediting:* For each edge, compute contribution of shortest paths from root to betweenness of that edge
  - Need to compute credits for nodes as well
  - Proceed from leaves to root, bottom-up

The Girvan-Newman Algorithm

Calculating Betweenness



BFS starting from *E*

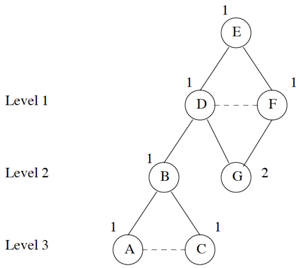
Adopted from `mmds.org`

Labeling Nodes

- Label each node by the number of shortest path to the root
- Start by labeling the root with 1
- Top-down, label each node by the sum of labels of each parents

THE GIRVAN-NEWMAN ALGORITHM

CALCULATING BETWEENNESS



BFS starting from E: Labeling

Adopted from `mmds.org`

EXAMPLE LABELING

- Label the *root* E with 1
- *Level 1:* Each D and F have only E as parent; label both with 1
- *Level 2:*
  - B has only D as parent, label with 1
  - G has parents D and F, label with 2
- *Level 3:* Both A, C have only B as parent, so both are labeled with 1

Crediting Nodes

- Compute fraction of shortest paths from root passing through node
- Credit each *leaf* with 1
  - If several shortest paths run to leaf, fractions add up to 1
- Each *non-leaf node v* gets credit

$$1 + \sum_{e \in \mathcal{D}(v)} c(e) \tag{1}$$

where $\mathcal{D}(v)$ are the DAG edges leaving from v, and $c(e)$ is the credit of an edge e

How to credit edges?

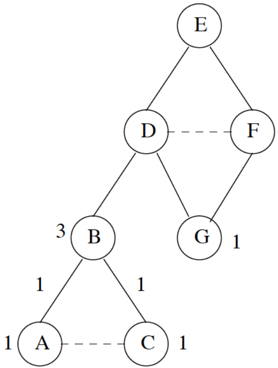
Crediting Edges

- Let $u_j, j = 1, ..., k$ be the parents of w; so (u_j, w) are the DAG edges entering w
- Let $N_j, j = 1, ..., k$ be the number of shortest paths from root running through edges (u_j, w)
- *Recall:* N_j agrees with the *label* of u_j, the number of shortest paths from root to u_j ...
- ... because every shortest path from root to u_j is a shortest path from root to w
- Let $c(w)$ be the credit of w
- We compute the credit of (u_i, w) as

$$c(u_i, w) := c(w) \times \frac{N_i}{\sum_{j=1}^{k} N_j} \tag{2}$$

The Girvan-Newman Algorithm

Calculating Betweenness



Crediting Nodes and Edges in Level 3 and 2

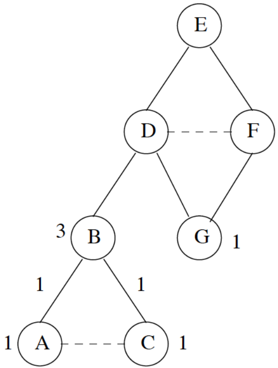
Adopted from `mmds.org`

Example Crediting

- *Level 3 Nodes:* Credit each of nodes A and C with 1
- *Level 2-3 Edges:* Both A and C have only one parent, so full credit 1 is assigned to both (B, A) and (B, C)

The Girvan-Newman Algorithm

Calculating Betweenness



Crediting Nodes and Edges in Level 3 and 2

Adopted from `mmds.org`

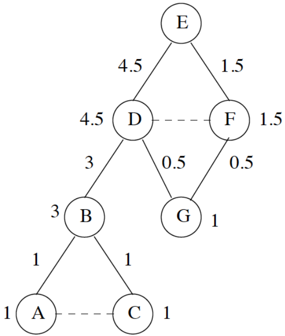
Example Crediting

Level 2 Nodes:

- G is a leaf, so gets credit 1
- B is not a leaf, so gets credit 1 + sum of credits 1 of DAG edges $(B, A), (B, C)$ leaving from it: credit 3 overall
- Intuitively, credit 3 for B refers to all shortest paths from E to A, B, C going through B.

THE GIRVAN-NEWMAN ALGORITHM

CALCULATING BETWEENNESS



Crediting Nodes and Edges

Adopted from `mmds.org`

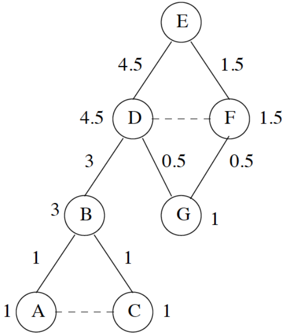
EXAMPLE CREDITING

Level 1-2 Edges:

- B has only one parent, D, so the edge (D, B) gets all of B's credit
- $(D, G), (F, G)$: Both D, F have label (not credit!) 1. So we credit both $(D, G), (F, G)$ with $1/(1 + 1) = 0.5$
- *Example:* If labels of D and F had been 3 and 5, the credit of (D, G) would be $3/(3 + 5) = 3/8$ and that of (F, G) would be $5/8$.

The Girvan-Newman Algorithm

Calculating Betweenness



Crediting Nodes and Edges

Adopted from `mmds.org`

Example Crediting

Level 1 Nodes / Edges:

- D gets credit 1 + credits of (D, B), (D, G) = credit 4.5 overall
- F gets credit 1 + credit of (F, G) = credit 1.5 overall
- Edges (E, D), (E, F) receive credits of D, F respectively, because D, F each have only one parent

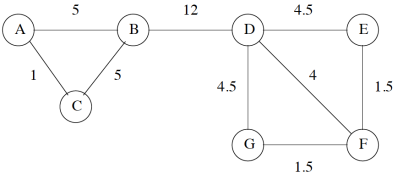
Summary: Credit on each edge is contribution to betweenness of that edge to shortest paths from E

THE GIRVAN-NEWMAN ALGORITHM

SUMMARY

COMPLETING THE ALGORITHM

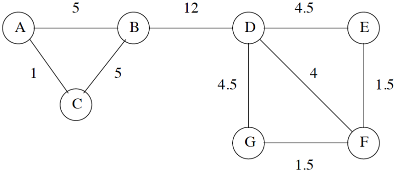
- Repeat the calculation illustrated for *E* for every other node
- Sum up the contributions for each edge across different roots
- Divide each edge weight by 2: each shortest path is counted twice, with each of its end points as root



Betweenness Scores

Adopted from `mmds.org`

Finding Communities with Betweenness



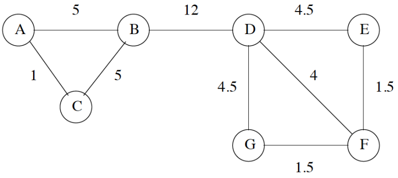
Betweenness Scores

Adopted from `mmds.org`

Computing Communities: Principle

- Remove edges in decreasing order of betweenness
- Stop at reasonably chosen threshold
- Communities are the resulting connected components

Finding Communities with Betweenness



Betweenness Scores

Adopted from `mmds.org`

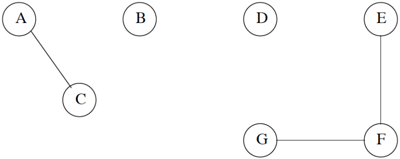
Computing Communities: Example Threshold 4

- First, remove (B, D): communities $\{A, B, C\}, \{D, E, F, G\}$
- Second, remove $(A, B), (B, C)$: communities $\{A, C\}, \{B\}, \{D, E, F, G\}$
- Third, remove $(D, E), (D, G)$: communities $\{A, C\}, \{B\}, \{D, E, F, G\}$
- Last, remove (D, F): communities $\{A, C\}, \{B\}, \{D\}, \{E, F, G\}$

Finding Communities with Betweenness

Computing Communities: Example Threshold 4

- First, remove (B, D): communities $\{A, B, C\}, \{D, E, F, G\}$
- Second, remove $(A, B), (B, C)$: communities $\{A, C\}, \{B\}, \{D, E, F, G\}$
- Third, remove $(D, E), (D, G)$: communities $\{A, C\}, \{B\}, \{D, E, F, G\}$
- Last, remove (D, F): communities $\{A, C\}, \{B\}, \{D\}, \{E, F, G\}$

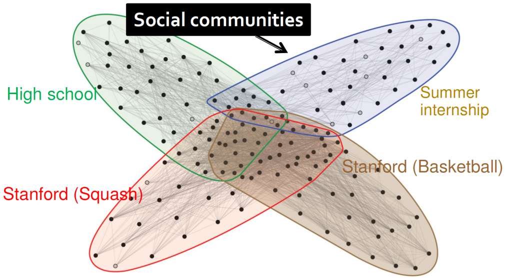


Final Communities

Adopted from `mmds.org`

The Graph Affiliation Model

Overlapping Communities

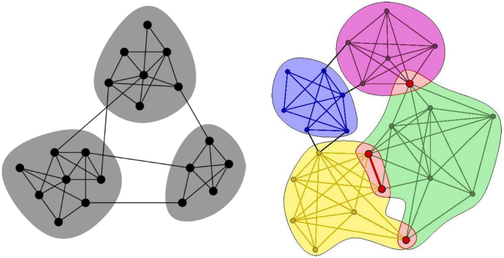


Subgraph from Facebook

Adopted from `mmds.org`

- *Observation:* Communities in social networks can overlap
- Graph partitioning does not help in these cases
- Would like to have a statistical interpretation of network data

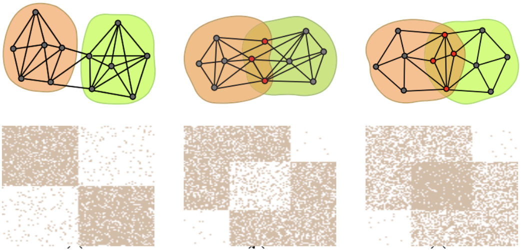
Nonoverlapping versus Overlapping Communities



Left: Nonoverlapping communities
Right: Overlapping communities
Adopted from `mmds.org`

- Communities may overlap or not
- *Issue:* How to determine communities correctly?

Affiliation Graph Model: Introduction

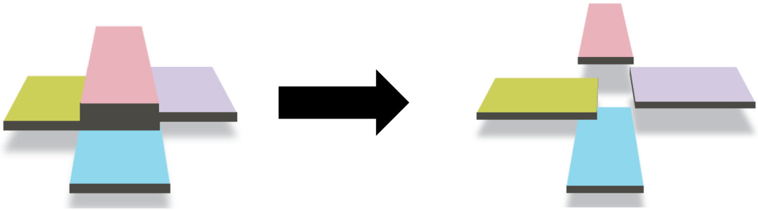


Networks and their adjacency matrices

Adopted from `mmds.org`

- Left: No overlap, adjacency matrix sparse across communities
- Middle: Loose overlap, adjacency matrix less sparse in shared part
- Right: Tight overlap, adjacency matrix dense in shared part

Community Discovery: Goal



Revealing (overlapping) communities

Adopted from `mmds.org`

- *Goal:* Discover communities correctly
- Regardless of whether they overlap or not

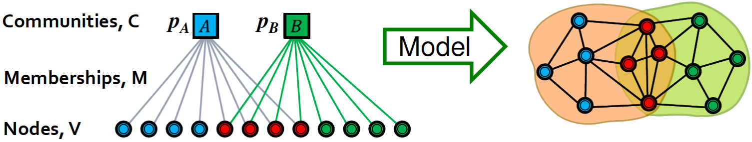
Determine the statistically most likely community structure

AFFILIATION GRAPH MODEL: INTRODUCTION

- *Issue:* Statistical control over community structure of a network
- *Idea:* Design *generative probability distribution*
- Given a number of nodes, this generative distribution generates edges
- The generative distribution represents a particular community structure
    - The distribution knows about nodes belonging to communities
    - It generates more edges within communities
    - It generates less edges between communities

Affiliation Graph Model: Introduction

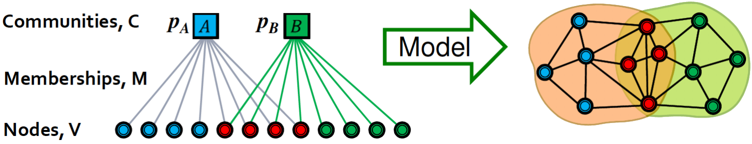
- The generative distribution represents community structures
  - The distribution knows about nodes belonging to communities
  - It generates more edges within communities
  - It generates less edges between communities



Distribution representing a community structure generating network

Adopted from `mmds.org`

Affiliation Graph Model: Introduction



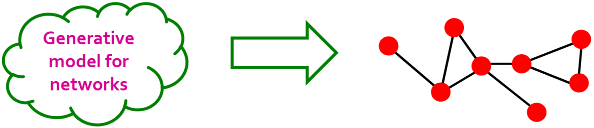
Distribution representing a community structure (left) generating network (right)

Adopted from `mmds.org`

- We can generate networks when knowing community structure
- *But:* We would like to determine the community structure when knowing the network

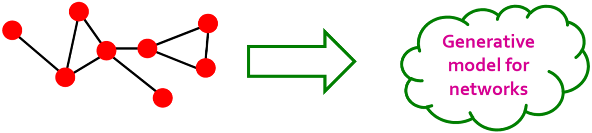
Isn't that exactly the opposite?

Generative Distributions



We can do this: generating network from distribution...

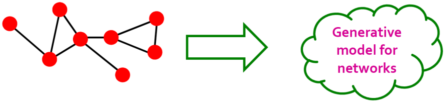
Adopted from `mmds.org`



...but we want this: inferring distribution from network

Adopted from `mmds.org`

Generative Distributions: Maximum Likelihood Inference



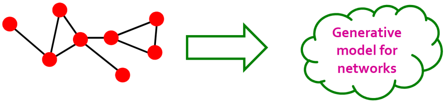
We want to infer distribution from network

Adopted from `mmds.org`

Maximum Likelihood Estimation

- Let Θ be a *parameterized class of probability distributions* that generate networks
  - We identify the different distributions with the different parameterizations
  - ☞ Formally not 100% correct, but doesn't matter here
- Let $\mathbf{P}(N \mid \theta)$ be the probability that distribution $\theta \in \Theta$ generates network N

Generative Distributions: Maximum Likelihood Inference



We want to infer distribution from network

Adopted from `mmds.org`

Maximum Likelihood Estimation

- Let $\mathbf{P}(N \mid \theta)$ be the probability that distribution $\theta \in \Theta$ generates network N
- *Maximum likelihood estimation:* Determine distribution $\hat{\theta}$ that generated N with greatest likelihood:

$$\hat{\theta} := \arg\max_{\theta \in \Theta} \mathbf{P}(N \mid \theta) \tag{3}$$

- This computes most reasonable distribution $\hat{\theta}$ for network N

Affiliation Graph Model: Definition I

- An AGM θ generates a network $N = (V, E)$ by adding edges E to a given set of nodes V
- For $u, v \in V$, edge (u, v) is generated with probability $\mathbf{P}_\theta((u, v))$
- $\mathbf{P}_\theta((u, v))$ depends on the parameters θ
- Recall that θ specifies community structure

So, what exactly is θ supposed to be?

Affiliation Graph Model: Parameters

- $\mathcal{C}$, as a set of *communities*
- $M \in \{0,1\}^{\mathcal{C} \times V}$, specifying *assignment of nodes $v \in V$ to communities* $C \in \mathcal{C}$, where

$$M_{C,v} = \begin{cases} 1 & v \text{ belongs to } C \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

  - M specifies "affiliations" of nodes $v \in V$
  - Note that one can vary $\mathcal{C}$, as a parameter, but not V
- $(p_C)_{C \in \mathcal{C}}$ as probabilities to generate edges (u, v) because $u, v \in C$
- *Summary:* A particular AGM θ corresponds to

$$\theta = (\mathcal{C}, M, (p_C)_{C \in \mathcal{C}}) \tag{5}$$

AFFILIATION GRAPH MODEL: $\mathbf{P}_\theta((u, v))$

Several C **containing both** u, v

- Let $M_u, M_v \subset \mathcal{C}$ be the subsets of communities that contain u and v, respectively
- Existence of communities that contain both u, v means

$$M_u \cap M_v \neq \emptyset$$

- Memberships in different communities have no influence on each other
- That is, we assume *statistical independence*

Affiliation Graph Model: $\mathbf{P}_\theta((u,v))$

Several C containing both u, v

- Statistical independence is expressed by

$$\prod_{C \in M_u \cap M_v} (1 - p_C)$$

  as probability of *no edge* (u, v) *in any community* $C \in M_u \cap M_v$
- Hence, the probability to generate (u, v) is

$$1 - \prod_{C \in M_u \cap M_v} (1 - p_C) \tag{6}$$

Done? No: What about $M_u \cap M_v = \emptyset$?

Affiliation Graph Model: $\mathbf{P}_\theta((u,v))$

No C containing both u, v

- For $M_u \cap M_v = \emptyset$, computing (6) yields (empty product is 1)

$$1 - \prod_{C \in \emptyset} (1 - p_C) = 1 - 1 = 0$$

- No edges across communities makes no sense
- Let $\epsilon > 0$ be small; we generate an edge (u, v) with probability

$$\mathbf{P}_\theta((u,v)) = \epsilon \quad \text{if} \quad M_u \cap M_v = \emptyset$$

Affiliation Graph Model: $\mathbf{P}_\theta((u,v))$

Affiliation Graph Model (AGM)

- An edge (u, v) is generated with probability

$$\mathbf{P}_\theta((u,v)) = \begin{cases} 1 - \prod_{C \in M_u \cap M_v}(1 - p_C) & M_u \cap M_v \neq \emptyset \\ \epsilon & M_u \cap M_v = \emptyset \end{cases} \tag{7}$$

- Edges (u, v) are generated independently from one another
- *Overall:* The probability $\mathbf{P}_\theta(E)$ to generate edges E given AGM θ computes as

$$\mathbf{P}_\theta(E) = \prod_{(u,v) \in E} \mathbf{P}_\theta((u,v)) \times \prod_{(u,v) \notin E} 1 - \mathbf{P}_\theta((u,v)) \tag{8}$$

where $\mathbf{P}_\theta((u,v))$ are computed following (7), with $\theta = (\mathcal{C}, M, p_C)$ determining p_C and M_u, M_v and so on.

Affiliation Graph Model: Overall Probability

Affiliation Graph Model (AGM)

- The probability $\mathbf{P}_\theta(E)$ to generate E given θ is

$$\mathbf{P}_\theta(E) = \prod_{(u,v)\in E} \mathbf{P}_\theta((u,v)) \times \prod_{(u,v)\notin E} 1 - \mathbf{P}_\theta((u,v)) \tag{9}$$

- *Reminder:* For a given network $N = (V, E)$, the *goal* is to determine

$$\hat{\theta} := \arg\max_{\theta\in\Theta} \mathbf{P}_\theta(E)$$

- That is, we need to vary $\theta = (\mathcal{C}, M, p_C)$ until $\mathbf{P}_\theta(E)$ is maximal

How to *systematically vary* $\theta = (\mathcal{C}, M, p_C)$?

Computing the MLE $\hat{\theta}$

Issues

- Search space of combinations of
  - Communities $\mathcal{C}$,
  - Assignments of nodes to communities M, and
  - Probabilities p_C for communities

  tends to be huge
- Concise formulas of (9) for $\mathbf{P}_\theta(E)$ as function of θ too difficult
- Analytical solution for determining $\hat{\theta} := \arg\max_{\theta \in \Theta} \mathbf{P}_\theta(E)$ not available
- Moreover, parameters are both discrete $(\mathcal{C}, M)$ and continuous $((p_C)_{C \in \mathcal{C}})$

Computing the MLE $\hat{\theta}$

Approach

1. Pick initial set of parameters θ_0
2. Vary θ such that $\mathbf{P}_\theta(E)$ iteratively increases
3. Vary $\mathcal{C}$ or M first
   - ☞ Partial derivates of $\mathbf{P}_\theta(E)$ wrt. p_C computable on fixed $\mathcal{C}, M$
4. Determine optimal $(p_C)_{C\in\mathcal{C}}$, e.g. by gradient descent
5. Keep change if $\mathbf{P}_\theta(E)$ has increased, discard otherwise

Computing the MLE $\hat{\theta}$

Iterative variations of $\mathcal{C}, M$

- *Varying M:*
  - Delete node from community, i.e. for $M_{C,v} = 1$, set $M_{C,v} = 0$
  - Add node to community, i.e. for $M_{C,v} = 0$, set $M_{C,v} = 1$
- *Varying $\mathcal{C}$:*
  - Merge two communities
  - Split community
  - Delete community
  - Add new community, with initial random selection of members

Computing the MLE $\hat{\theta}$

Soft Community Membership

- Instead of $M_{C,v} \in \{0,1\}$, allow any real-numbered $M_{C,v} \geq 0$
- For (u,v) to be generated because of $u,v \in C$, let

$$\mathbf{P}_\theta((u,v)) = 1 - e^{-M_{C,u}M_{C,v}} \tag{10}$$

  be the individual probability
- Proceeding exactly as before, we obtain

$$\mathbf{P}_\theta(E) = \prod_{(u,v)\in E}(1 - e^{-\sum_C M_{C,u}M_{C,v}}) \prod_{(u,v)\notin E} e^{-\sum_C M_{C,u}M_{C,v}} \tag{11}$$

Computing the MLE $\hat{\theta}$

Soft Community Membership

- Probability for edges E:

$$\mathbf{P}_\theta(E) = \prod_{(u,v)\in E} (1 - e^{-\sum_C M_{C,u} M_{C,v}}) \prod_{(u,v)\notin E} e^{-\sum_C M_{C,u} M_{C,v}} \tag{12}$$

- On fixed communities, include M in gradient descent (or related) optimization step
- *Advantages:*
  - Only one gradient descent run necessary
  - Less prone to get stuck in unfavorable local optima
- If necessary, add or delete communities, and re-run

Direct Discovery of Overlapping Communities

Introduction

- *Popular idea:* Determine communities as (induced) subgraphs of a certain type
- Subgraphs should contain unusually large amount of edges
- Subgraphs are allowed to overlap
- Will treat two types briefly here:
  - Cliques
  - Complete bipartite subgraphs

FINDING CLIQUES

DEFINITION [INDUCED SUBGRAPH]

Let $G = (V, E)$ be a graph. A subgraph $C = (V' \subset V, E' \subset E)$ is *induced* iff

$$(v', w') \in E \quad \text{implies} \quad (v', w') \in E'$$

for any $v', w' \in V'$.

DEFINITION [CLIQUE]

Let $G = (V, E)$ be a graph.

- An induced subgraph $C = (V', E')$ is called a *clique* iff any pair of nodes in C is connected by an edge.
- A clique $C = (V', E')$ is *maximal* iff extending the clique by any node and its edges implies that the clique property no longer holds.

Communities as Cliques

- *Possible idea:* Determine communities as maximal cliques
- *Caveat:* The number of maximal cliques in a graph may be exponential in the number of nodes
- So, listing all maximal cliques is a computationally demanding problem
- Nevertheless, identifying communities as clique like arrangements is popular

Complete Bipartite Graphs

Definition [(Complete) Bipartite Graphs]

A graph $G = (V, E)$ with vertices V and edges E is referred to as *bipartite* iff

- there are $V_1, V_2 \subset V$ such that

$$V = V_1 \,\dot{\cup}\, V_2 \quad \text{and} \quad E \subset (V_1 \times V_2)$$

- A bipartite graph $G = (V, E)$ is *complete* iff

$$V = V_1 \,\dot{\cup}\, V_2 \quad \text{and} \quad E = (V_1 \times V_2)$$

  that is iff each node from V_1 is connected with each node from V_2

- A complete bipartite graph where $|V_1| = s, |V_2| = t$ is referred to as $K_{s,t}$
- A complete bipartite graph is also referred to as *biclique*

Complete Bipartite Graphs and Communities

- *Strategy:* Seek to discover all sufficiently large bicliques
- Treat them as "nuclei" (or seeds) of communities
- *Theoretical Advantage over Cliques:* While it is not possible to guarantee the existence of large cliques for graphs with many edges, one can guarantee the existence of large bicliques

Finding Complete Bipartite Graphs

Frequent Itemset Mining Problem

- Let $G = (V, E)$ on $V = V_1 \dot{\cup} V_2$ be a (large) bipartite graph
- Items are nodes from V_1
- Baskets are nodes from V_2
- Items in baskets are nodes from V_1 connected to basket node
- $K_{s,t}$ in G is itemset of size s that appears in t baskets
- So mining for frequent itemsets at threshold t dicovers all $K_{s,t}$

General / Further Reading

Literature

- Mining Massive Datasets, Sections 10.1, 10.2, 10.3, 10.5 `http://infolab.stanford.edu/~ullman/mmds/ch10.pdf`
- Next lecture: "Web Advertisements": sections 8.1 – 8.4 in *Mining of Massive Datasets*