Mining Massive Datasets:

Week 6: Communities in Social Networks

Ian Quah (itq)

June 16, 2017

Topic 1

Community Detection in Graphs

None

Topic 2

Affiliation Graph Model

1. **Plan**
   (a) Given a model, generate a network
   (b) Given a network, find the "best" model
2. **Model of Networks**
   (a) Goal: Define a model that can generate networks
   Model will have a set of "parameters" that will later want to estimate (and detect communities)
   (b) Solution: Community-Affiliation Graph
      i. A generative model B(V, C, M, $\{p_c\}$) for graphs:
         A. Nodes, V
         B. Communities, C
         C. Memberships, M
         D. Each community c has a single probability p_c
         E. Later, we fit model to networks to detect communities
      ii. **AGM generates Links for each pair of nodes**
         A. For each pair of nodes in community A, connect them with prob p_A
         B. The overall edge probability is:

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

where
· M_u is the set of communities node u belongs to
· If u, v share no communities: P(u, v) = ϵ
· "OR" function: if at least 1 community says yes, create an edge
Intuition
(1 - p_c) is the prob that both of them do not connect
$\prod$ is essentially saying all communities are voting "no", there is no connection
      iii. **AGM characteristics**
         A. Can express: non-overlapping, overlapping, nested

Topic 3

AGM → BIGCLAM

Main idea: AGM is restrictive: it models a "Yes-No" relationship
Relaxation: Memberships have strengths

1. F_{UA}: The membership strength of node u to community A (F_{uA} = 0: no membership)

2. **Probability of two nodes belonging to the same community**

   Each community A links nodes independently:

   P_A(u, v) = 1 - exp(-F_{uA} · F_{vA})

3. **Membership across multiple groups**

   Matrix F describes our membership matrix

   - single col describes the strength of a single entity belonging in some community

   - single row describes the membership of all nodes in that single community

   Prob. that at least one common community C links the nodes:

$$P(u, v) = 1 - \prod_{C} (1 - P_C(u, v))$$

4. **AGM → BIGCLAM**

   Take Equation from (2) and equation from (3)

$$\begin{aligned} P_A(u, v) &= 1 - exp(-F_{uA} \cdot F_{vA}) && \text{(Equation from 2)} \\ P(u, v) &= 1 - \prod_{C} (1 - P_C(u, v)) && \text{(General form of equation from 2)} \\ &= 1 - exp(-\sum_{C} F_{uC} F_{vC}) \\ &= 1 - exp(-F_u \cdot F_v^T) \end{aligned}$$

Topic 4

Solving the BIGCLAM

1. **Problem**: Now we need to estimate F, the community affiliation matrix.

   Solution: Find F that maximizes the likelihood

$$argmax_F \prod_{(u,v) \in E} p(u, v) \prod_{(u,v) \notin E} (1 - p(u, v))$$

   where p(u, v) = 1 - exp(-$F_u \cdot F_v^T$)

   If we work with the log-likelihood instead, then we obtain:

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

2. **New Problem**: How do we run the optimization?

   Solution: Gradient Descent, which leaves us with the following equation

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

where $\mathcal{N}(u)$ is the set of outgoing neighbors for u

Coordinate gradient ascent

(a) Iterate over rows of F

(b) Compute gradient ∇l(F_u) of row u while keeping others fixed

(c) Update row F_u: $F_u \leftarrow F_u + \eta\nabla l(F_u)$

(d) Project F_u back to a non-negative vector: If $F_{uC} < 0$: $F_{uC} = 0$

But, this is slow because it is linear: we need to iterate through all the data because of the summations

3. **BigCLAM: 2.0:**

   Notice: $\sum_{v \notin \mathcal{N}(u)} F_v = (\sum F_v - F_u - \sum_{v \in \mathcal{N}(u)} F_v)$

   (a) We can cache $\sum_v F_v$

   Thus, computing $\sum_{v \notin \mathcal{N}(u)} F_v$ now takes linear time in the degree of $|\mathcal{N}(u)|$ of u (instead of linear in the data)

   In networks degree of a node is much smaller to the total number of nodes in the network, so this is a significant speedup

Note, the next few topics are considered advanced, and are focused on cluster detection in large graphs

Topic 5

Detecting Communities as Clusters

None

Topic 6

What makes a Good cluster?

1. Given an undirected graph, G(V,E)

2. **Partitioning Task**

   Divide vertices into 2 disjoint groups, A, B = V\A and B = V\A

3. **But how do we define a "Good" cluster in G?**

   Evaluation Metrics

   (a) Maximize number of within-cluster connections

   (b) Minimize number of between cluster connections

   Graph Cuts

   (a) Express cluster quality as a function of the "edge cut" of the cluster

   (b) Cut: Set of edges with only 1 node in the cluster

$$cut(A) = \sum_{i \in A, j \notin A} W_{ij}$$

   which works for weighted and unweighted graphs

(c) Can be thought of as number of outgoing edges from the cluster

(d) **Problem:**

i. Degenerate cases: doesn't always do what we want.

ii. Only considers external cluster connections

iii. Does not consider internal cluster connectivity

Gives rise to our next criteria: Conductance

(a) Conductance: connectivity of group to rest of network relative to density of group, ϕ

$$\phi(A) = \frac{|\{(i,j) \in E; i \in A, j \notin A\}|}{min(vol(A), 2m - vol(A))}$$

Numerator: the cut

Denominator: total weight of edges with at least one endpoint in A:

$$vol(A) = \sum_{i \in A} d_i$$

(b) Produces balanced clusters

Topic 7

Graph Laplacian Matrix

1. Finding a good partition is NP-hard

2. **Spectral Graph Partitioning**

(a) Let A: adjacency matrix of undirected graph, G
A_{ij} = 1 if (i, j) is an edge, else 0

(b) x is a vector in $\mathcal{R}^n$ with components (x_1, ..., x_n)
Think of it as a label/ value of each node of G

(c) **What is the meaning of A $\cdot$ x?**

$$y_i = \sum_{i,j}^{n} A_{ij} x_j = \sum_{i,j \in E} x_j$$

Intuition: y_i is a sum of labels x_j of neighbors of i

(d) **Spectral Graph Theory**

i. Analyze the "spectrum" of matrix representing G

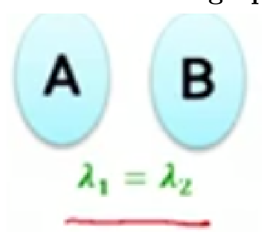
ii. **Spectrum:** Eigenvectors x_i of a graph ordered by magnitude of their corresponding eigenvalues λ_i

Topic 8

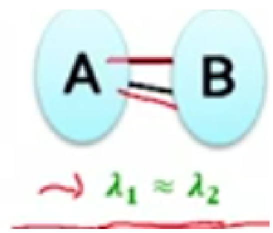
Examples of Eigendecompositions of Graphs

1. **D-regular graphs**
   (a) Suppose all nodes in G have degree d, and G is connected
   (b) What are some e-vals/vecs of G?
   A $\cdot$ x = $\lambda \cdot$ x — What is λ? What is x?
   If we let x = $[1]_n$, then we get $\lambda = [\mathrm{d}]_n$

2. **Graph on 2 separate components**
   (a) **G not connected?**
      i. What are some eigenvectors?
      ii. In this case, we have two cases of x: x' and x", s.t x' = [1, 1, ..., 0, 0] where it is 1 when describing elements in A, and 0 when describing elements in B. The opposite is true for x".
      iii. Then, we have x' = [1,1,...,0,0] then A $\cdot$ x' = [d,d,...,0,0] and x" = [0,0,...,d,d]
   (b) **But what if the graph is badly connected?**



Disconnected
$\lambda_1 = \lambda_2$
Serves as an estimate



Badly Connected
$\lambda_1 \simeq \lambda_2$