Kronecker graphs and the Structure of Large Networks

CS224W: Social and Information Network Analysis
Jure Leskovec, Stanford University
http://cs224w.stanford.edu
Recap: Network Community Profile

Best community has ~100 nodes

Communities get worse and worse

Better and better communities

$\Phi(k)$, (conductance)

$k_r$ (cluster size)
Explanation: Nested core-periphery

Denser and denser network core

Small good communities

Nested core-periphery
Idea: Recursive graph generation

- Intuition: self-similarity leads to power-laws
- Try to mimic recursive graph / community growth
- There are many obvious (but wrong) ways:

  - **Kronecker Product** is a way of generating self-similar matrices
Kronecker product: Graph

Intermediate stage

Adjacency matrix

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 1 \\
0 & 1 & 1 \\
\end{bmatrix}
\]

(3x3)

\[
K_1
\]

Adjacency matrix

\[
\begin{bmatrix}
K_1 & K_1 & 0 \\
K_1 & K_1 & K_1 \\
0 & K_1 & K_1 \\
\end{bmatrix}
\]

(9x9)

\[
K_2 = K_1 \otimes K_1
\]
Kronecker product: Definition

- Kronecker product of matrices $A$ and $B$ is given by

$$C = A \otimes B = \begin{pmatrix}
    a_{1,1}B & a_{1,2}B & \ldots & a_{1,m}B \\
    a_{2,1}B & a_{2,2}B & \ldots & a_{2,m}B \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{n,1}B & a_{n,2}B & \ldots & a_{n,m}B \\
\end{pmatrix}_{N*K \times M*L}$$

- Define a Kronecker product of two graphs as a Kronecker product of their adjacency matrices.
Kronecker graphs

- **Kronecker graph**: a growing sequence of graphs by iterating the Kronecker product

\[ K_1^{[k]} = K_k = \left( K_1 \otimes K_1 \otimes \ldots \otimes K_1 \right) \quad k \text{ times} \]

- Each Kronecker multiplication exponentially increases the size of the graph
- \( K_k \) has \( N_1^k \) nodes and \( E_1^k \) edges, so we get **densification**
- One can easily use multiple initiator matrices (\( K_1', K_1'', K_1''' \)) that can be of different sizes
Continuing multiplying with $K_1$ we obtain $K_4$ and so on…

$K_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

$K_2 = K_1 \otimes K_1 = \begin{bmatrix} K_1 & K_1 & 0 \\ K_1 & K_1 & K_1 \\ 0 & K_1 & K_1 \end{bmatrix}$

$K_4$ adjacency matrix

$3 \times 3$ matrix

$9 \times 9$ matrix
Kronecker initiator matrices

Initiator $K_1$  

$K_1$ adjacency matrix  

$K_3$ adjacency matrix
Kronecker graphs have many properties found in real networks:

- **Properties of static networks**
  - Power-Law like Degree Distribution
  - Power-Law eigenvalue and eigenvector distribution
  - Small Diameter

- **Properties of dynamic networks**
  - Densification Power Law
  - Shrinking/Stabilizing Diameter
Theorem: Constant diameter: If $G_l$ has diameter $d$ then graph $G_k$ also has diameter $d$

Observation: Edges in Kronecker graphs:

$\text{Edge } (X_{i,j}, X_{k,l}) \in G \otimes H$

iff $(X_i, X_k) \in G$ and $(X_j, X_l) \in H$

where $X$ are appropriate nodes

Example:

$d(X_{i,j}, X_{k,l}) = \max\{d(X_i, X_k), d(X_j, X_l)\}$

Central node is $X_{2,2}$
Stochastic Kronecker graphs

\[ \Theta_k = (2^k, I \cdot I^k) \]

- Create \( N_1 \times N_1 \) probability matrix \( \Theta_1 \) with \( 0 < \theta_{ij} < 1 \).
- Compute the \( k^{th} \) Kronecker power \( \Theta_k \).
- For each entry \( p_{uv} \) of \( \Theta_k \), include an edge \((u, v)\) in \( K_k \) with probability \( p_{uv} \).

### Instance

<table>
<thead>
<tr>
<th></th>
<th>0.25</th>
<th>0.10</th>
<th>0.10</th>
<th>0.04</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.05</td>
<td>0.15</td>
<td>0.02</td>
<td>0.06</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>0.02</td>
<td>0.15</td>
<td>0.06</td>
</tr>
<tr>
<td>0.3</td>
<td>0.01</td>
<td>0.03</td>
<td>0.03</td>
<td>0.09</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kronecker multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Theta_1 )</td>
</tr>
<tr>
<td>( N_1 = 2 )</td>
</tr>
<tr>
<td>( E_1 = 1.1 )</td>
</tr>
</tbody>
</table>

\[ \Theta_2 = \Theta_1 \times \Theta_1 \]

Flip biased coins
What is known about Stochastic Kronecker?

- **Undirected** Kronecker graph model with:
  - Connected, if:
    - \( b+c > 1 \)
  - Connected component of size \( \Theta(n) \), if:
    - \( (a+b)(b+c) > 1 \)
  - Constant diameter, if:
    - \( b+c > 1 \)
  - Not searchable by a decentralized algorithm
Given a real network $G$

Want to estimate initiator matrix:

- **Method of moments** [Owen ‘09]
  - Compare counts of and solve the system of equations

- **Maximum likelihood** [ICML ‘07]
  - $\arg \max P( | \Theta_1 )$

- **SVD** [VanLoan-Pitsianis ‘93]
  - Can solve $\min ||G - \Theta_1 \otimes \Theta_2||_F^2$ using SVD
Kronecker graphs: Estimation

- Maximum likelihood estimation

\[ \arg \max_{\Theta_1} P(\Theta_1 | \Theta_0) \]

- Naïve estimation takes \( O(N!N^2) \):
  - \( N! \) for different node labelings:
    - **Our solution:** Metropolis sampling: \( N! \rightarrow (\text{big}) \text{ const} \)
  - \( N^2 \) for traversing graph adjacency matrix
    - **Our solution:** Kronecker product \( (E \ll N^2): N^2 \rightarrow E \)

- Do gradient descent
Maximum likelihood estimation

- Given real graph $G$
- Find Kronecker initiator graph $\Theta$ (i.e., \begin{array}{cc} a & b \\ c & d \end{array})
  which
  $$\arg \max_{\Theta} P(G \mid \Theta)$$
- We need to (efficiently) calculate
  $$P(G \mid \Theta)$$
- And maximize over $\Theta$
  (e.g., using gradient descent)
KronFit: Likelihood $P(G | \Theta)$

- Given a graph $G$ and Kronecker matrix $\Theta$ we calculate probability that $\Theta$ generated $G$.

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

| $\Theta$ | $\Theta_k$ | $P(G | \Theta)$ |
|----------|------------|-----------------|
| 0.5 0.2  | 0.25 0.10  |                |
| 0.1 0.3  | 0.05 0.15  |                |
|          | 0.05 0.02  |                |
|          | 0.01 0.03  |                |
| 1 0 1 1  | 0 1 1 0 0  |                |
| 1 0 1 1  | 1 0 1 1 1  |                |
| 1 1 1 1  | 1 1 1 1 1  |                |
**Challenge 1: Node correspondence**

- **Nodes are unlabeled**
- **Graphs** $G'$ and $G''$ should have the same probability $P(G' | \Theta) = P(G'' | \Theta)$
- One needs to consider all node correspondences $\sigma$

\[
P(G' | \Theta) = \sum_\sigma P(G' | \Theta, \sigma) P(\sigma)
\]

- **All correspondences are a priori equally likely**
- **There are $O(N!)$ correspondences**
Assume we solved the correspondence problem.

Calculating

$$P(G | \Theta) = \prod_{(u,v) \in G} \Theta_k \left[ \sigma_u, \sigma_v \right] \prod_{(u,v) \notin G} (1 - \Theta_k \left[ \sigma_u, \sigma_v \right])$$

Takes $O(N^2)$ time.

Infeasible for large graphs ($N \sim 10^5$)

$$p(\xi | \Theta) = p(\neg \xi | \Theta)$$

$$\sum_{(v,w) \in E} -p(n(w)) + p(\neg n(w))$$
Solution 1: Node correspondence

- Log-likelihood
  \[ l(\Theta) = \log \sum_{\sigma} P(G|\Theta, \sigma)P(\sigma) \]

- Gradient of log-likelihood
  \[ \frac{\partial}{\partial \Theta} l(\Theta) = \sum_{\sigma} \frac{\partial \log P(G|\sigma, \Theta)}{\partial \Theta} P(\sigma|G, \Theta) \]

- Sample the permutations from \( P(\sigma|G, \Theta) \) and average the gradients
Solution 1: Node correspondence

- Metropolis sampling:
  - Start with a random permutation \( \sigma \)
  - \( \sigma' = \) swap two elements in permutation \( \sigma \)
  - Accept the new permutation \( \sigma' \)
    - If new permutation is better (gives higher likelihood)
    - Else accept with prob. proportional to the ratio of likelihoods
      (no need to calculate the normalizing constant!)

\[
\frac{P(\sigma' | G, \Theta)}{P(\sigma | G, \Theta)}
\]
Sampling node labelings (2)

\[ \sigma^{(0)} := (1, \ldots, N) \]

repeat

Draw \( j \) and \( k \) uniformly from \((1, \ldots, N)\)

\[ \sigma^{(i)} := \text{SwapElements}(\sigma^{(i-1)}, j, k) \]

Draw \( u \) from \( U(0,1) \)

if \( u > \frac{P(\sigma^{(i)}|G, \Theta)}{P(\sigma^{(i-1)}|G, \Theta)} \) then

\[ \sigma^{(i)} := \sigma^{(i-1)} \]

end if

\( i = i + 1 \)

until \( \sigma^{(i)} \sim P(\sigma|G, \Theta) \)

return \( \sigma^{(i)} \)

- Need to efficiently calculate the likelihood ratios
- But the permutations \( \sigma^{(i)} \) and \( \sigma^{(i+1)} \) only differ at 2 positions
- So we only traverse to update 2 rows (columns) of \( \Theta_k \)
- We can evaluate the likelihood ratio efficiently

Metropolis permutation sampling algorithm

\[ \sigma^{(0)} := (1, \ldots, N) \]

\[ \sigma^{(i)} := \text{SwapElements}(\sigma^{(i-1)}, j, k) \]

\[ i = i + 1 \]

\[ \text{until } \sigma^{(i)} \sim P(\sigma|G, \Theta) \]

\[ \text{return } \sigma^{(i)} \]
Solution 2: Calculating $P(G|\Theta, \sigma)$

- Calculating naively $P(G|\Theta, \sigma)$ takes $O(N^2)$
- Idea:
  - First calculate likelihood of empty graph, a graph with 0 edges
  - Correct the likelihood for edges that we observe in the graph
- By exploiting the structure of Kronecker product we obtain closed form for likelihood of an empty graph
Solution 2: Calculating $P(G|\Theta, \sigma)$

- We approximate the likelihood:

$$l(\Theta) \approx l_e(\Theta) + \sum_{(u,v) \in G} - \log(1 - \Theta_k[\sigma_u, \sigma_v]) + \log(\Theta_k[\sigma_u, \sigma_v])$$

  - Empty graph
  - No-edge likelihood
  - Edge likelihood

- The sum goes only over the edges
- Evaluating $P(G|\Theta, \sigma)$ takes $O(E)$ time
- Real graphs are sparse, $E << N^2$
Real graphs are sparse so we first calculate likelihood of empty graph

- Probability of edge $(i, j)$ is in general $p_{ij} = \theta_1^a \theta_2^b \theta_3^c \theta_4^d$
- By using Taylor approximation to $p_{ij}$ and summing the multinomial series we obtain:

$$l_e(\Theta) = \sum_{i,j=1}^{N} \log(1 - p_{ij}) \approx -\left( \sum_{i,j=1}^{N_1} \theta_{i,j} \right)^k - \frac{1}{2} \left( \sum_{i,j=1}^{N_1} \theta_{i,j}^2 \right)^k$$

Taylor approximation

$$\log(1-x) \sim -x - 0.5 x^2$$

- We approximate the likelihood:

$$l(\Theta) \approx l_e(\Theta) + \sum_{(u,v) \in G} -\log(1 - \Theta_k[\sigma_u, \sigma_v]) + \log(\Theta_k[\sigma_u, \sigma_v])$$

Empty graph

No-edge likelihood

Edge likelihood
Experiments: real networks

- Experimental setup
  - Given real graph $G$
  - Gradient descent from random initial point
  - Obtain estimated parameters $\Theta$
  - Generate synthetic graph $K$ using $\Theta$
  - Compare properties of graphs $G$ and $K$

- Note:
  - We do not fit the graph properties themselves
  - We fit the likelihood and then compare the properties
Can gradient descent recover true parameters?

- Generate a graph from random parameters
- Start at random point and use gradient descent
- We recover true parameters 98% of the times
Real and Kronecker are very close: 

\[ \Theta_1 = \begin{pmatrix} 0.99 & 0.54 \\ 0.49 & 0.13 \end{pmatrix} \]
Real and Kronecker are very close:  
$$\Theta_1 = \begin{pmatrix} 0.99 & 0.57 \\ 0.51 & 0.22 \end{pmatrix}$$  

**Figure:**

(a) In-Degree

(b) Out-degree

(c) Triangle participation

(d) Hop plot

(e) Scree plot

(f) “Network” value
What do estimated parameters tell us about the network structure?

\[ K_1 = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \]
What do estimated parameters tell us about the network structure?

\[
K_1 = \begin{pmatrix}
0.9 & 0.5 \\
0.5 & 0.1
\end{pmatrix}
\]

Nested Core-periphery
Small and large networks are very different:

\[ K_1 = \begin{pmatrix} 0.99 & 0.17 \\ 0.17 & 0.82 \end{pmatrix} \]
Implications (1)

- Large scale network structure:
  - Large networks are different from small networks and manifolds
  - Nested Core-periphery
    - Recursive onion-like structure of the network where each layer decomposes into a core and periphery
Implications (2)

- Remember the SKG theorems:
  - Connected, if $b+c > 1$:
    - $0.55 + 0.15 > 1$. No!
  - Giant component, if $(a+b) \cdot (b+c) > 1$:
    - $(0.99 + 0.55) \cdot (0.55 + 0.15) > 1$. Yes!

- Real graphs are in the parameter region analogous to the giant component of an extremely sparse $G_{np}$
Kronecker model: Alternative view

- Each node $u$ has associated binary vector $A_u$
  - Think of it as feature vector
- Initiator matrix $K$ acts like a "similarity" matrix
- Probability of a link between nodes $u$, $v$:

$$P(u, v) = \prod_{i=1}^{k} K_1(A_u(i), A_v(i))$$

$$K_1 = \begin{bmatrix} 0 & 1 \\ a & b \\ c & d \end{bmatrix}$$

$v_2 = (0,1)$
$v_3 = (1,0)$

$$P(v_2, v_3) = b \cdot c$$

$$K_2 = K_1 \otimes K_1$$
For each node \( u \) we have a binary vector \( A_u \)

For each edge \((u,v)\) determine prob.

\[
P(u,v) = \prod_{i=1}^{k} K_i(A_u(i), A_v(i))
\]

\[
A_u = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0
\end{bmatrix}
\]

\[
A_v = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0
\end{bmatrix}
\]

\[
K_i = \begin{bmatrix}
\alpha_1 & \beta_1 & \alpha_2 & \beta_2 \\
\beta_1 & \gamma_1 & \beta_2 & \gamma_2 \\
\alpha_3 & \beta_3 & \alpha_4 & \beta_4 \\
\beta_3 & \gamma_3 & \beta_4 & \gamma_4
\end{bmatrix}
\]

\[
P(u,v) = \alpha_1 \cdot \beta_2 \cdot \gamma_3 \cdot \alpha_4
\]
MAG: Interpretation

- How to think of

\[ K_i = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \]

\[ \frac{A_u}{A_v} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \]

\[ K_i = \begin{bmatrix} \alpha_1 & \beta_1 \\ \beta_1 & \gamma_1 \end{bmatrix} \]

\[ P(u,v) = \alpha_1 \cdot \beta_2 \cdot \gamma_3 \cdot \alpha_4 \]

- Attribute-attribute similarity matrix:
  - Can model homophily:
    \[ \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \]
  - Heterophily:
    \[ \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix} \]
  - Core-periphery:
    \[ \begin{pmatrix} 0.9 & 0.5 \\ 0.5 & 0.1 \end{pmatrix} \]
Simplified MAG model

- For each node $u$ generate a binary vector $A_u$
  - draw $k$ ($k \approx \log_2(|V|)$) independent samples from a Bernoulli($\lambda$)

- For each pair of nodes $(u,v)$ determine an edge prob.

$$P(u,v) = \prod_{i=1}^{k} K(A_u(i), A_v(i))$$
2 ingredients of Kronecker model:

- (1) Each of $2^k$ nodes has a unique binary vector of length $k$
  - Node id expressed binary is the vector
- (2) The initiator matrix $K$

Question:

- What if ingredient (1) is dropped?
  - i.e., do we need high variability of feature vectors?
Comparison: Adjacency matrices

Adjacency matrices:

\[ A_U = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \]
Coalitions in signed networks

- Received 19 entries
- Top score: 14,690
- Top 10:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Score</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wang,Fan</td>
<td>0</td>
<td>10%</td>
</tr>
<tr>
<td>2</td>
<td>Cui,Jingyu</td>
<td>+1</td>
<td>10%</td>
</tr>
<tr>
<td>3</td>
<td>Preston,Dan</td>
<td>+3</td>
<td>8%</td>
</tr>
<tr>
<td>4</td>
<td>Pham,Peter Thien Tan</td>
<td>+4</td>
<td>8%</td>
</tr>
<tr>
<td>5</td>
<td>Wang,Dakan</td>
<td>+9</td>
<td>6%</td>
</tr>
<tr>
<td>6</td>
<td>Moreinis,Stanislav</td>
<td>+13</td>
<td>4%</td>
</tr>
<tr>
<td>7</td>
<td>Wang,Chunyan</td>
<td>+17</td>
<td>2%</td>
</tr>
<tr>
<td>8</td>
<td>Kim,Hyung Jin</td>
<td>+19</td>
<td>2%</td>
</tr>
<tr>
<td>9</td>
<td>Wu,Yu</td>
<td>+28</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Jin,Ye</td>
<td>+31</td>
<td></td>
</tr>
</tbody>
</table>

Random partitioning gives score of ~50,000
(1) Everyone used some form of greedy hill-climbing:

- Repeat until no improvement:
  - (1) Pick a node (multiple nodes), move it to the other side if it improves a score
  - (2) Pick an edge, move the endpoints so that score is most improved

(2) Randomization techniques and simulated annealing to escape local minima

- Repeat (1) until no improvement
- Randomize and restart

(3) Signed Laplacian matrix