Stanford CS224W: Generative Models for Graphs

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu
Motivation for Graph Generation

- So far, we have been **learning from graphs**
  - We assume the graphs are given

- **But how are these graphs generated?**

---

Social Networks  Economic Networks  Communication Networks

Image credit: Medium  Image credit: Science  Image credit: Lumen Learning
We want to generate realistic graphs, using graph generative models.
Why Do We Study Graph Generation

- **Insights** – We can understand the formulation of graphs
- **Predictions** – We can predict how the graph will further evolve
- **Simulations** – We can use the same process to generate novel graph instances
- **Anomaly detection** - We can decide if a graph is normal/abnormal
Road Map of Graph Generation

- **Step 1**: Properties of real-world graphs
  - A successful graph generative model should fit these properties

- **Step 2**: Traditional graph generative models
  - Each come with different assumptions on the graph formulation process

- **Step 3**: Deep graph generative models
  - Learn the graph formulation process from the data
  - Cover in the next lecture
Stanford CS224W: Properties of Real-world Graphs
We will characterize graphs by:

Degree distribution: $P(k)$

Clustering coefficient: $C$

Connected components: $s$

Path length: $h$

We have introduced these notions in Lecture 1&2. Here we give a quick recap.
(1) Degree Distribution

- **Degree distribution** $P(k)$: Probability that a randomly chosen node has degree $k$
  
  $N_k = \#$ nodes with degree $k$

- Normalized histogram:
  
  $$P(k) = \frac{N_k}{N} \rightarrow \text{plot}$$
(2) Clustering Coefficient

- **Clustering coefficient:**
  - How connected are $i$'s neighbors to each other?
  - Node $i$ with degree $k_i$
  - $C_i = \frac{2e_i}{k_i(k_i - 1)}$
    where $e_i$ is the number of edges between the neighbors of node $i$

- **Graph clustering coefficient:**
  - Take average over all the nodes
    $$C = \frac{1}{N} \sum_{i}^N C_i$$

$C_i \in [0, 1]$
(3) Connectivity

- **Size of the largest connected component**
  - Largest set where any two vertices can be joined by a path
- **Largest component = Giant component**

How to find connected components:
- Start from random node and perform Breadth First Search (BFS)
- Label the nodes that BFS visits
- If all nodes are visited, the network is connected
- Otherwise find an unvisited node and repeat BFS
(4) Path Length

- **Diameter**: The maximum (shortest path) distance between any pair of nodes in a graph.

- **Average path length** for a connected graph or a strongly connected directed graph:

  \[ \bar{h} = \frac{1}{2E_{\text{max}}} \sum_{i,j \neq i} h_{ij} \]

  - \( h_{ij} \) is the distance from node \( i \) to node \( j \)
  - \( E_{\text{max}} \) is the max number of edges (total number of node pairs) = \( n(n-1)/2 \)

- Many times we compute the average only over the connected pairs of nodes (that is, we ignore “infinite” length paths).
Case Study: MSN Graph

- **MSN Messenger:**
- **1 month of activity**
  - 245 million users logged in
  - 180 million users engaged in conversations
  - More than 30 billion conversations
  - More than 255 billion exchanged messages
Network: 180M people, 1.3B edges
MSN: (1) Degree Distribution

Degree, k

Count, \( P(k) \times n \)

3.5e+007
3e+007
2.5e+007
2e+007
1.5e+007
1e+007
5e+006
0
0 2000 4000 6000 8000 10000
Note: We plotted the same data as on the previous slide, just the axes are now logarithmic.
MSN: (2) Clustering

Avg. clustering of the MSN:

\[ C = 0.1140 \]

\[ C_k: \text{average } C_i \text{ of nodes } i \text{ of degree } k: \]

\[ C_k = \frac{1}{N_k} \sum_{i:k_i=k} C_i \]
MSN: (3) Connected Components

[Graph showing the distribution of weakly connected component sizes with a logarithmic scale. The x-axis represents the weakly connected component size, and the y-axis represents the count. The largest component contains 99.9% of the nodes.]
MSN: (4) Path Length

Number of links between pairs of nodes in the largest connected component

Avg. path length 6.6
90% of the nodes can be reached in < 8 hops

<table>
<thead>
<tr>
<th>Steps</th>
<th>#Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>78</td>
</tr>
<tr>
<td>3</td>
<td>3,96</td>
</tr>
<tr>
<td>4</td>
<td>8,648</td>
</tr>
<tr>
<td>5</td>
<td>3,299,252</td>
</tr>
<tr>
<td>6</td>
<td>28,395,849</td>
</tr>
<tr>
<td>7</td>
<td>79,059,497</td>
</tr>
<tr>
<td>8</td>
<td>52,995,778</td>
</tr>
<tr>
<td>9</td>
<td>10,321,008</td>
</tr>
<tr>
<td>10</td>
<td>1,955,007</td>
</tr>
<tr>
<td>11</td>
<td>518,410</td>
</tr>
<tr>
<td>12</td>
<td>149,945</td>
</tr>
<tr>
<td>13</td>
<td>44,616</td>
</tr>
<tr>
<td>14</td>
<td>13,740</td>
</tr>
<tr>
<td>15</td>
<td>4,476</td>
</tr>
<tr>
<td>16</td>
<td>1,542</td>
</tr>
<tr>
<td>17</td>
<td>536</td>
</tr>
<tr>
<td>18</td>
<td>167</td>
</tr>
<tr>
<td>19</td>
<td>71</td>
</tr>
<tr>
<td>20</td>
<td>29</td>
</tr>
<tr>
<td>21</td>
<td>16</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
</tr>
</tbody>
</table>
Degree distribution: Heavily skewed; avg. degree = 14.4

Clustering coefficient: 0.11

Connectivity: giant component

Path length: 6.6

Are these values “expected”? Are they “surprising”? To answer this, we need a model!
Stanford CS224W: Erdös-Renyi Random Graphs
Simplest Model of Graphs

- **Erdös-Renyi Random Graphs** [Erdös-Renyi, ‘60]
- Two variants:
  - $G_{np}$: undirected graph on $n$ nodes where each edge $(u,v)$ appears i.i.d. with probability $p$
  - $G_{nm}$: undirected graph with $n$ nodes, and $m$ edges picked uniformly at random

What kind of networks do such models produce?
Random Graph Model $G_{np}$

- $n$ and $p$ do not uniquely determine the graph!
  - The graph is a result of a random process
  - We can have many different realizations given the same $n$ and $p$

$n = 10$
$p = 1/6$
Properties of $G_{np}$

Degree distribution: $P(k)$

Clustering coefficient: $C$

Path length: $h$

What are the values of these properties for $G_{np}$?
(1) Degree Distribution of $G_{np}$

- **Fact:** Degree distribution of $G_{np}$ is **binomial**.
- Let $P(k)$ denote the fraction of nodes with degree $k$:

$$P(k) = \binom{n-1}{k} p^k (1 - p)^{n-1-k}$$

- Mean, variance of a binomial distribution

$$\bar{k} = p(n - 1)$$

$$\sigma^2 = p(1 - p)(n - 1)$$
(2) Clustering Coefficient of $G_{np}$

Remember: \[ C_i = \frac{2e_i}{k_i(k_i - 1)} \]

Edges in $G_{np}$ appear i.i.d. with prob. $p$

So, expected $E[e_i]$ is: \[ E[e_i] = p \frac{k_i(k_i - 1)}{2} \]

Each pair is connected with prob. $p$

Number of distinct pairs of neighbors of node $i$ of degree $k_i$

Then $E[C_i]$: \[ E[C_i] = p \frac{k_i(k_i - 1)}{k_i(k_i - 1)} = p = \frac{k}{n - 1} \approx \frac{\bar{k}}{n} \]

Clustering coefficient of a random graph is small. If we generate bigger and bigger graphs with fixed avg. degree $k$ (that is we set $p = k \cdot 1/n$), then $C$ decreases with the graph size $n$. 

Where $e_i$ is the number of edges between $i$’s neighbors.
(3) Connected Components of $G_{np}$

- **Graph structure of $G_{np}$ as $p$ changes:**

  
<table>
<thead>
<tr>
<th>$p$</th>
<th>Avg deg = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Empty graph</td>
</tr>
<tr>
<td>1/(n-1)</td>
<td>Giant component appears</td>
</tr>
<tr>
<td>c/(n-1)</td>
<td>Avg. deg const. Lots of isolated nodes.</td>
</tr>
<tr>
<td>log(n)/(n-1)</td>
<td>Fewer isolated nodes.</td>
</tr>
<tr>
<td>2*log(n)/(n-1)</td>
<td>No isolated nodes.</td>
</tr>
</tbody>
</table>

- **Emergence of a giant component:**
  - avg. degree $k=2E/n$ or $p=k/(n-1)$
    - **Degree $k=1-\varepsilon$:** all components are of size $\Omega(\log n)$
    - **Degree $k=1+\varepsilon$:** 1 component of size $\Omega(n)$, others have size $\Omega(\log n)$
      - Each node has at least one edge in expectation
$G_{np}$ Simulation Experiment

$G_{np}, n=100,000, k=p(n-1) = 0.5 \ldots 3$

Fraction of nodes in the largest component

$p^*(n-1) = 1$
Network Properties of $G_{np}$

Degree distribution:
\[ P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k} \]

Clustering coefficient:
\[ C = p = \frac{\bar{k}}{n} \]

Connectivity:
GCC exists when $k > 1$.

Path length:

next!
Def: Expansion

- Graph $G(V, E)$ has **expansion** $\alpha$: if $\forall S \subseteq V$:
  
  # of edges leaving $S \geq \alpha \cdot \min(|S|, |V \setminus S|)$

- Or equivalently:

  $$\alpha = \min_{S \subseteq V} \frac{\# \text{edges leaving } S}{\min(|S|, |V \setminus S|)}$$
Expansion: Measures Robustness

- **Expansion is measure of robustness:**
  - To disconnect $l$ nodes, we need to cut $\geq \alpha \cdot l$ edges

- **Low expansion:**

- **High expansion:**

- **Social networks:**
  - “Communities”

\[ \alpha = \min_{s \in \mathcal{F}} \frac{\# \text{edges leaving } S}{\min(|S|, |V\setminus S|)} \]
**Expansion: Random Graphs**

- **Fact:** In a graph on $n$ nodes with expansion $\alpha$ for all pairs of nodes there is a path of length $O((\log n)/\alpha)$.

- **Random graph $G_{np}$:**
  For $\log n > np > c$, $\text{diam}(G_{np}) = O(\log n / \log (np))$
  - Random graphs have good expansion so it takes a logarithmic number of steps for BFS to visit all nodes.
(4) Shortest Path of $G_{np}$

Erdös-Renyi Random Graph can grow very large but nodes will be just a few hops apart

Shortest Path length follows $O(\log n)$

Here we control the average degree to be constant
Back to MSN vs. $G_{np}$

**Degree distribution:**
- **MSN:**
- **$G_{np}:**

**Avg. path length:**
- **MSN:**
- **$G_{np}:**

**Avg. clustering coef.:**
- **MSN:**
- **$G_{np}:**

**Largest Conn. Comp.:**
- **MSN:**
- **$G_{np}:**
Real Networks vs. $G_{np}$

- **Are real networks like random graphs?**
  - Giant connected component: 😊
  - Average path length: 😊
  - Clustering Coefficient: 😞
  - Degree Distribution: 😞

- **Problems with the random networks model:**
  - Degree distribution differs from that of real networks
  - Giant component in most real networks does NOT emerge through a phase transition
  - No local structure – clustering coefficient is too low

- **Most important: Are real networks random?**
  - The answer is simply: **NO!**

2/25/21

Jure Leskovec, Stanford CS224W: Machine Learning with Graphs, cs224w.stanford.edu
Stanford CS224W: The Small-World Model

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu
Motivation for Small-World

Can we have **high clustering** while also having **short paths**?

### MSN vs. \( G_{np} \)

<table>
<thead>
<tr>
<th></th>
<th>MSN</th>
<th>( G_{np} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. path length:</td>
<td>6.6</td>
<td>( O(\log n) )</td>
</tr>
<tr>
<td>Avg. clustering coef.:</td>
<td>0.11</td>
<td>( \bar{k} / n )</td>
</tr>
</tbody>
</table>

\( h \approx 8.2 \)
\( C \approx 8 \cdot 10^{-8} \)

- **Regular lattice graph:**
  - High clustering coefficient
  - High diameter

- **\( G_{np} \) random graph:**
  - Low clustering coefficient
  - Low diameter
Real networks have high clustering:
- MSN network has $7$ orders of magnitude larger clustering than the corresponding $G_{np}$!

Other examples:

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Degree</th>
<th>$h_{\text{actual}}$</th>
<th>$h_{\text{random}}$</th>
<th>$C_{\text{actual}}$</th>
<th>$C_{\text{random}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Film actor collaborations</td>
<td>225,226</td>
<td>61.00</td>
<td>3.65</td>
<td>2.99</td>
<td>0.79</td>
<td>0.00027</td>
</tr>
<tr>
<td>Power Grid</td>
<td>4,941</td>
<td>2.67</td>
<td>18.70</td>
<td>12.40</td>
<td>0.080</td>
<td>0.005</td>
</tr>
<tr>
<td>C. elegans</td>
<td>282</td>
<td>14.00</td>
<td>2.65</td>
<td>2.25</td>
<td>0.28</td>
<td>0.05</td>
</tr>
</tbody>
</table>

"actual" ... real network
"random" ... random graph with same avg. degree
The “Controversy”

- **Consequence of expansion:**
  - **Short paths:** $O(\log n)$
    - This is the smallest diameter we can get if we keep the degree constant.
  - But clustering is low!
- **But networks have “local” structure:**
  - **Triadic closure:**
    - Friend of a friend is my friend
  - High clustering but diameter is also high
- How can we have both?

$G_{np}$ random graph:
- Low clustering coefficient
- Low diameter

Regular lattice graph:
- High clustering coefficient
- High diameter
**Small-world Graphs: Idea**

- **Idea:** Interpolate between regular lattice graphs and $G_{np}$ random graph

- **How do we interpolate between these two graphs?**

Regular lattice graph:  
*High clustering coefficient*  
*High diameter*

Interpolate

Small-world graph:  
*High clustering coefficient*  
*Low diameter*

$G_{np}$ random graph:  
*Low clustering coefficient*  
*Low diameter*
Solution: The Small-World Model

- Small-World Model
- Two components to the model:
  - (1) Start with a *low-dimensional regular lattice*
    - (In our case we are using a ring as a lattice)
    - Has high clustering coefficient

Solution: The Small-World Model

- Small-World Model
- Two components to the model:
  - (2) **Rewire:** Introduce randomnessness ("shortcuts")
    - Add/remove edges to create shortcuts to join remote parts of the lattice
    - For each edge, with prob. $p$, move the other endpoint to a random node
The Small-World Model

Rewiring allows us to “interpolate” between a regular lattice and a random graph.

\[ h = \frac{N}{2k} \quad C = \frac{1}{2} \]

\[ h = \frac{\log N}{\log \alpha} \quad C = \frac{\bar{k}}{N} \]

[D. Wattts and S. Strogatz, Collective dynamics of ‘small-world’networks, Nature]
The Small-World Model

Parameter region of high clustering and low path length

Intuition: It takes a lot of randomness to ruin the clustering, but a very small amount to create shortcuts.
Could a network with high clustering be at the same time a small world?

- Yes! You don’t need more than a few random links

The Small-World Model:

- Provides insight on the interplay between clustering and the small-world
- Captures the structure of many realistic networks
- Accounts for the high clustering of real networks
- Does not lead to the correct degree distribution
Stanford CS224W: Kronecker Graph Model

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu
How can we think of network structure recursively? **Intuition:** Self-similarity

- **Object is similar to a part of itself:** the whole has the same shape as one or more of the parts

**Mimic recursive graph/community growth:**

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

- Kronecker graphs:
  - A recursive model of network structure

\[
K_1 \quad K_1 \quad 0 \\
K_1 \quad K_1 \quad K_1 \\
0 \quad K_1 \quad K_1
\]

\[K_2 = K_1 \otimes K_1\]

3 x 3 9 x 9 81 x 81 adjacency matrix
Kronecker Product: Definition

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

$$C = A \otimes B = \begin{pmatrix}
    a_{1,1}B & a_{1,2}B & \ldots & a_{1,m}B \\
    a_{2,1}B & a_{2,2}B & \ldots & a_{2,m}B \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{n,1}B & a_{n,2}B & \ldots & a_{n,m}B
\end{pmatrix}
$$

$N \times M \quad K \times L$

$N*K \times M*L$

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

- **Kronecker graph** is obtained by growing sequence of graphs by iterating the **Kronecker product** over the initiator matrix $K_1$:

\[
K_1^{[m]} = K_m = K_1 \otimes K_1 \otimes \ldots K_1 = K_{m-1} \otimes K_1
\]

\(m\) times

- **Note:** One can easily use multiple initiator matrices ($K_1', K_1'', K_1'''$) (even of different sizes)
Kronecker Initiator Matrices

Initiator $K_1$

$K_1$ adjacency matrix

$K_3$ adjacency matrix
**Stochastic Kronecker Graphs**

- **Step 1**: Create $N_1 \times N_1$ probability matrix $\Theta_1$
- **Step 2**: Compute the $k^{th}$ Kronecker power $\Theta_k$
- **Step 3**: For each entry $p_{uv}$ of $\Theta_k$ include an edge $(u, v)$ in $K_k$ with probability $p_{uv}$

\[ \Theta_2 = \Theta_1 \otimes \Theta_1 \]

<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>0.1</td>
<td>0.01</td>
<td>0.03</td>
<td>0.03</td>
<td>0.09</td>
</tr>
</tbody>
</table>

**Kronecker multiplication**

**Probability of edge** $p_{uv}$

**Instance matrix** $K_2$

**Flip biased coins**
How do we generate an instance of a (Directed) stochastic Kronecker graph?

Is there a faster way? YES!

Idea: Exploit the recursive structure of Kronecker graphs

“Drop” edges onto the graph one by one

Need to flip $n^2$ coins!! Way too slow!!
A faster way to generate Kronecker graph

How to “drop” an edge into a graph $G$ on $n=2^m$ nodes
Generation of Kronecker Graphs

- A faster way to generate Kronecker graph

\[
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\]

- How to “drop” an edge into a graph G on \( n=2^m \) nodes
Generation of Kronecker Graphs

- A faster way to generate Kronecker graph

\[ \Theta = \begin{array}{cc}
  a & b \\
  c & d
\end{array} \]

- How to “drop” an edge into a graph \( G \) on \( n=2^m \) nodes

Adjacency matrix \( G \)
A faster way to generate Kronecker graph

How to “drop” an edge into a graph $G$ on $n=2^m$ nodes:

We may get a few edges colliding. We simply reinsert them.
Fast Kronecker generator algorithm:
- For generating directed graphs
- **Insert 1 edge on graph G on** \( n = 2^m \) **nodes:**
  - Create normalized matrix \( L_{uv} = \Theta_{uv} / (\sum_{op} \Theta_{op}) \)
  - **For** \( i = 1 \ldots m \)
    - Start with \( x = 0, y = 0 \)
    - Pick a row/column \( (u, v) \) with prob. \( L_{uv} \)
    - Descend into quadrant \( (u, v) \) at level \( i \) of \( G \)
      - **This means:** \( x += u \cdot 2^{m-i}, y += v \cdot 2^{m-i} \)
    - Add an edge \( (x, y) \) to \( G \)
**Estimation: Epinions (n=76k, m=510k)**

- **Real** and **Kronecker** are very close:

\[
\Theta_1 = \begin{pmatrix}
0.99 & 0.54 \\
0.49 & 0.13
\end{pmatrix}
\]

- **Diagram:**
  - (a) In-Degree
  - (b) Out-degree
  - (c) Triangle participation
  - (d) Hop plot
  - (e) Scree plot
  - (f) "Network" value
Today: Traditional graph generative models

- Erdös-Renyi graphs
- Small-world graphs
- Kronecker graphs

All these models have prior assumption of the graph generation processes

Next: Deep graph generative models

Learn the graph generation process from raw data