NetGAN
Generating Graphs via Random Walks

Aleksandar Bojchevski*, Oleksandr Shchur*, Daniel Zügner*, Stephan Günnemann
Technical University of Munich, Germany

ICML 2018

* Equal contribution
Generative models for graphs

Graphs are ubiquitous
  • Social networks
  • Protein-protein interaction networks
  • Knowledge graphs

Generative models for graphs
  • Imputation of missing values (link prediction, recommendation)
  • Sampling / data simulation (planning, augmentation)
  • Controllable generation
  • Representation learning

## Generative models for graphs

**What makes a good generative model for graphs?**

Should capture essential observed patterns of real-world graphs!

<table>
<thead>
<tr>
<th></th>
<th>Sparsity</th>
<th>Power law degree dist.</th>
<th>Community structure</th>
<th>Triangle counts</th>
<th>Homophily</th>
<th>Unknown patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planted Part.</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>???</td>
</tr>
<tr>
<td>DC-SBM</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>???</td>
</tr>
<tr>
<td>ERGM</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>???</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>???</td>
</tr>
<tr>
<td>Super model</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>???</td>
</tr>
</tbody>
</table>
Generative models for graphs

What makes a good generative model for graphs?
Should capture essential observed patterns of real-world graphs!

<table>
<thead>
<tr>
<th></th>
<th>Sparsity</th>
<th>*Power law degree dist.</th>
<th>Community structure</th>
<th>Triangle counts</th>
<th>#Homophily</th>
<th>Unknown patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planted Part.</td>
<td>X</td>
<td>?</td>
<td>X</td>
<td>???</td>
<td></td>
<td>???</td>
</tr>
<tr>
<td>DC-SBM</td>
<td>X</td>
<td>?</td>
<td>X</td>
<td>???</td>
<td></td>
<td>???</td>
</tr>
<tr>
<td>ERGM</td>
<td>X</td>
<td>?</td>
<td>X</td>
<td>???</td>
<td></td>
<td>???</td>
</tr>
<tr>
<td>...</td>
<td>X</td>
<td>?</td>
<td>X</td>
<td>???</td>
<td></td>
<td>???</td>
</tr>
<tr>
<td>Super model</td>
<td>X</td>
<td>?</td>
<td>X</td>
<td>X</td>
<td>???</td>
<td>???</td>
</tr>
</tbody>
</table>

* Broido & Clauset 2018, #Dong et al. 2017
Generative models for graphs

What makes a good generative model for graphs?

How do we define a model that captures all the essential (potentially still unknown) properties of real graphs?

Learn them from data ⇒ Implicit models

Super model X ??? X X ??? ???

*Broido & Clauset 2018, #Dong et al. 2017
Explicit vs. implicit models

• **Explicit** models have a parametric specification of the data distribution
  - Observe patterns and manually specify a model to capture them
  - Learn via MLE, MAP, ...

• **Implicit** models define a stochastic process that directly generates data
  - Likelihood free: learn by comparison with the true data distribution (e.g. class probability estimation, GANs)

\begin{align*}
\theta & \quad \begin{tikzpicture}
    \node [draw, circle] (z) at (0,0) {$z_i$};
    \node [draw, above of=z, circle] (x) at (0,1) {$x_i$};
    \draw [-stealth] (z) -- (x);
\end{tikzpicture} \\
\mathcal{L} & = \log p(X | \theta)
\end{align*}

\begin{align*}
\mathcal{L} & = \frac{p^*}{q_\theta}
\end{align*}

Setting and challenges

1. Single large graph as input
   • Compared to e.g. many images in computer vision

2. Quadratic scaling and sparsity
   • For $N$ nodes there are $N^2$ possible edges
   • Real graphs have $|E| \ll N^2$ significantly fewer edges

3. Discrete output samples
   • Can’t easily backpropagate through sampling step

4. Permutation invariance
Tackling the challenges

Our solution: learn a distribution of random walks over the graph
NetGAN architecture

NetGAN generator

$z \sim \mathcal{N}(0, I_d)$

$m_0 = g_{\theta'}(z)$

$(p_t, m_t) = f_{\theta}(m_{t-1}, v_{t-1})$

$v_t \sim \text{Cat}(\sigma(p_t))$

Tackling discreteness:
Gumbel-Softmax straight-through estimator (Jang et al., 2017)
Graph assembly: sample edges with probability proportional to their transition counts
Random walk length

Random walks are a Markov process – they don’t have memory.

What’s the benefit of having length greater than 2?

Helps with generalization (empirically)
Evaluation and goals

Unlike images we cannot visually inspect large graphs for quality

Goal 1) Fidelity - capture known graph patterns without manually specifying them

Goal 2) Generalization – capture the general topology
- Avoid trivial solution of memorizing training graph
- Impute missing data

Goal 3) Latent space interpolation
- Yields graphs with smoothly changing properties

What can we do with NetGAN?

Generate graphs that have similar structure but are not replicas.
Reproduce known patterns

... without manually specifying them

<table>
<thead>
<tr>
<th>Model</th>
<th>Max degree</th>
<th>Power law exp.</th>
<th>Intra-Com. density</th>
<th>Inter-Com. density</th>
<th>Clustering coefficient</th>
<th>Character. path len.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original graph</td>
<td>240</td>
<td>1.86</td>
<td>1.7e-3</td>
<td>4.3e-4</td>
<td>2.7e-3</td>
<td>5.61</td>
</tr>
<tr>
<td>Configuration model</td>
<td>240</td>
<td>1.86</td>
<td>2.8e-4</td>
<td>1.6e-3</td>
<td>3.0e-4</td>
<td>4.38</td>
</tr>
<tr>
<td>DC-SBM</td>
<td>165</td>
<td>1.81</td>
<td>1.2e-3</td>
<td>6.7e-4</td>
<td>3.3e-3</td>
<td>5.12</td>
</tr>
<tr>
<td>ERGM</td>
<td>243</td>
<td>1.79</td>
<td>1.2e-3</td>
<td>6.9e-4</td>
<td>2.2e-3</td>
<td>4.59</td>
</tr>
<tr>
<td>BTER</td>
<td>199</td>
<td>1.79</td>
<td>7.5e-4</td>
<td>1.0e-3</td>
<td>4.6e-3</td>
<td>4.59</td>
</tr>
<tr>
<td>VGAE</td>
<td>13</td>
<td>1.67</td>
<td>3.2e-4</td>
<td>1.4e-3</td>
<td>1.2e-3</td>
<td>5.28</td>
</tr>
<tr>
<td>NetGAN</td>
<td>233</td>
<td>1.79</td>
<td>1.4e-3</td>
<td>6.0e-4</td>
<td>2.4e-3</td>
<td>5.20</td>
</tr>
</tbody>
</table>
Generalize – impute missing data

Idea: assess link prediction performance on held out data via transition counts
Higher transition count implies that the edge is more likely

<table>
<thead>
<tr>
<th>number of nodes</th>
<th>Cora-ML 2.8 K</th>
<th>Citeseer 2.1 K</th>
<th>Pubmed 19.7 K</th>
<th>PolBlogs 1.8 K</th>
<th>DBLP 16.2 K</th>
<th>Cora 18.8 K</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of edges</td>
<td>7.9 K</td>
<td>3.7 K</td>
<td>44.3 K</td>
<td>16.7 K</td>
<td>51.9 K</td>
<td>64.5 K</td>
</tr>
<tr>
<td>Adamic/Adar</td>
<td>92.16</td>
<td>88.69</td>
<td>84.98</td>
<td>85.43</td>
<td>91.13</td>
<td>93.00</td>
</tr>
<tr>
<td>DC-SBM</td>
<td><strong>96.03</strong></td>
<td>94.77</td>
<td><strong>96.76</strong></td>
<td>95.46</td>
<td><strong>97.05</strong></td>
<td>98.01</td>
</tr>
<tr>
<td>node2vec</td>
<td>92.19</td>
<td>95.29</td>
<td>96.49</td>
<td>85.10</td>
<td>96.41</td>
<td><strong>98.52</strong></td>
</tr>
<tr>
<td>VGAE</td>
<td>95.79</td>
<td>95.11</td>
<td>94.50</td>
<td>93.73</td>
<td>96.38</td>
<td>97.59</td>
</tr>
<tr>
<td>NetGAN (500K)</td>
<td>94.00</td>
<td>95.18</td>
<td>87.39</td>
<td>95.06</td>
<td>82.45</td>
<td>82.31</td>
</tr>
<tr>
<td>NetGAN (100M)</td>
<td>95.19</td>
<td><strong>96.30</strong></td>
<td>93.41</td>
<td><strong>95.51</strong></td>
<td><strong>86.61</strong></td>
<td><strong>84.82</strong></td>
</tr>
</tbody>
</table>
Interpolate between graphs

Instead of sampling from the entire 2D latent space, sample from small sub-regions
Each pixel in the plots corresponds to a graph (statistic) sampled from a sub-region

\[ \sim \sim \mathcal{N}(0, I_2) \]

Claw count
Gini coefficient
Rel. edge dist. entropy
Interpolate between graphs

The density of communities also changes smoothly
Limitations and future work

Fixed graph size
• Can only generate graphs of same size as the input

Scalability
• Larger graphs need more random walks to get representative transition counts

Evaluation
• How to evaluate the quality beyond graph statistics

Other settings: heterogenous, attributed, multi-graph, ....
Summary

NetGAN – Deep implicit model generating graphs via random walks

Captures (un?)known graph patterns without manually specifying them
Generalizes as shown by its link prediction performance
Generates graphs with smoothly changing properties

Code: github.com/danielzuegner/netgan
Poster: #58, Hall B, Today