Last week:
- unsupervised learning
  - clustering nodes into groups based on the graph
- semi-supervised learning
  - a few nodes labeled try to classify the rest
- today: representation learning
  node embeddings
Construct function $f$

$$f: V \rightarrow \mathbb{R}^k$$

$k < \frac{n}{10}$

low-dimensional embedding

sparse data $\Rightarrow$ compress to a dense representation that preserves structure

Why?

- Convenient for "downstream" ML tasks

Today:
1. Hoff et al. latent space clustering
2. Henderson et al. RoleX
   (role of nodes)
3. node2vec (Grover &Leskovec)

Example: spectral clustering
Downstream: embedding convenient for k-means

Latent Space Models (Hoff et al. '02)

z_1, ..., z_n be latent positions
of \( n \) nodes in a network \( z_i \in \mathbb{R}^k \)

Idea: nearby nodes in the latent space are likely to connect

\[
\forall (i, j) \in E
\]

\[
0 \rightarrow g \left( \left\| z_i - z_j \right\| \right)
\]

\[
= \frac{1}{1 + \exp(\beta \left\| z_i - z_j \right\|^2 + \alpha)}
\]

Makes sense for two social reasons

1. Reciprocity
Link \( i \rightarrow j \)
\( \Rightarrow \) \( z_i, z_j \) likely close
\( \Rightarrow j \rightarrow i \) is likely
\( \Rightarrow \) reciprocation is common

2 links \( i \rightarrow j, i \rightarrow k \)
\( \Rightarrow \) \( z_i, z_j \) likely close
\( z_i, z_k \) likely close
\( \Rightarrow \) \( z_j, z_k \) likely close
\( \Rightarrow j \) and \( k \) likely to link
\( \Rightarrow \) triangles / clustering
Optimize with maximum likelihood

Observe graph $A$

$$\max_{\{z_i\}} \Pr(A | z_1, \ldots, z_n)$$

$$\Pr( (i,j) \in E )$$

$$= \frac{1}{1 + \exp(\beta (z_i - z_j)^2 + \lambda)}$$

$$\prod \Pr( (i,j) | z_i, z_j )$$

$$\prod (1 - \Pr( (i,j) | z_i = z_j )$$
Still about clustering (similar to spectral but probabilistic)

What else?

RolX (Henderson et al. ’12)

Alternative clustering approach
Collect features about nodes

- degree
- # triangles
- PageRank
- squared degree
Key idea: features of nodes

Cluster in feature space

Dimensionality reduction of the data matrix

- PCA
- Robust PCA
- NMF

Problem: how to manually pick out features
Problem: clustering vs. roles

node2vec
(Grover + Jure Leskovec 2016)

\[
\max_{\{z_u\}} \sum_u \log \left( \Pr(N_S(u) \mid z_u) \right)
\]

Embedding
Sampled neighborhood

1) Conditional independence
\[
\Pr(N_S(u) \mid z_u) = \prod_{v \in N_S(u)} \Pr(v \mid z_u)
\]
2 Softmax probabilities

\[ P(v | z_u) \]

\[ = \frac{\exp(z_v^T z_u)}{\sum \exp(z_w^T z_u) \text{ for } w \neq u} \]

Opt. problem

\[ \max \left \{ -\log Y_u + \sum_{v \in \text{Ns}(u)} z_{uw} \right \} \]

\[ Y_u = \sum_{w \neq u} \exp(z_w^T z_u) \]

Computing \( Y_u \) can be expensive
Use "negative sampling" approximating $Y_u$ with a small random sample $W \subseteq V$.

Remaining part is how we actually get $N_s(u)$?

Two sampling strategies:

1. BFS
2. DFS

Basic idea: run BFS or DFS for a little while (truncate)
also take multiple samples

BFS - nearby nodes first

captures homophily (similar nodes are friends), similar to clustering

DFS - captures more macro-scale network properties closer to role discovery?

Actual strategy

Start: \( \circ \xrightarrow{w} \bigcirc \)
with prob. $p_i$ return to $w$

with prob. $q_i$ BFS

A lot of hyperparameters $p, q$ control BFS/DFS
also how long "walks" or samples should be
also how many samples per node?

Last week/this week:
ML for graphs
Rest of networks:
1. Small subgraph patterns
2. Centrality/ranking graphs & hypergraphs