March 21, 2019

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

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

$$k \ll n = |V|$$

low-dimensional embedding

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

Why?

- Convenient for "downstream" ML tasks

Today:

'02 ① Hoff et al. latent space
→ clustering
'12 ② Henderson et. al. RolX
→ role of nodes

'16 ③ node2vec (Grover + Leskovec)
→ ① + ②

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

Latent Space Models (Hoff et al. '02)

$z_1, \ldots, z_n$ 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

$$(i) \rightarrow (j)$$

$$\Pr((i,j) \in E) = \frac{1}{1 + \exp(\beta \|z_i - z_j\|^2 + \alpha)}$$

Makes sense for two social reasons

① Reciprocity

Link $i \to j$

$\Rightarrow$ z_i, z_j likely close

$\Rightarrow$ $j \to i$ is likely

$\Rightarrow$ reciprocation is common

2 links $i \to j$, $i \to k$

$\Rightarrow$ 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

$$\max_{\{z_i\}} \Pr(A \mid z_1, \dots, z_n)$$

$$\Pr((i,j) \in E) = \frac{1}{1 + \exp(\beta \|z_i - z_j\|^2 + \alpha)}$$

$$\prod_{(i,j) \in E} \Pr((i,j) \mid z_i, z_j) \cdot \prod_{(i,j) \notin E} \left(1 - \Pr((i,j) \mid z_i, z_j)\right)$$

Still about clustering (similar to spectral but probabalistic)

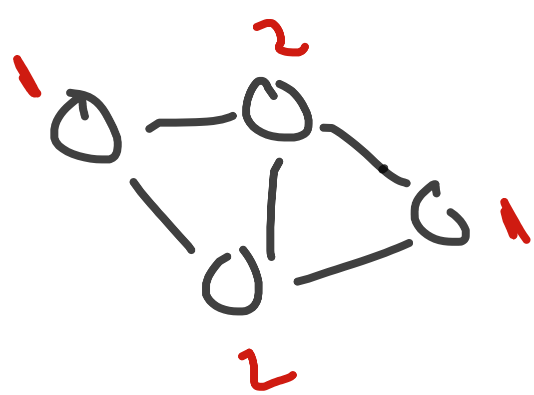
What else?

RolX (Henderson et al. '12)

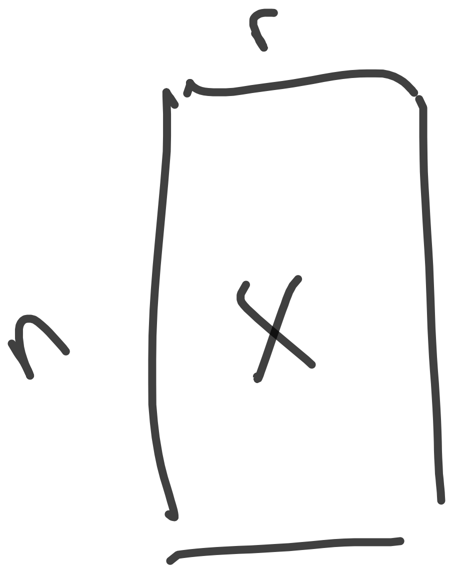
Alternative clustering approach

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: have to manually pick out features

Problem: clustering vs. roles

---

node2vec

(Grover + Jure Leskovec 2016)

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

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

- $\Pr(v \mid z_u)$

$$= \frac{\exp(z_v^T z_u)}{\sum_{w \neq u} \exp(z_w^T z_u)}$$

Opt. problem

$$\max_{\{z_u\}} \sum_u \left[-\log Y_u + \sum_{v \in N_S(u)} z_v^T z_u \right]$$

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

Computing Y_u ~~is~~ can be expensive

Use "negative sampling"

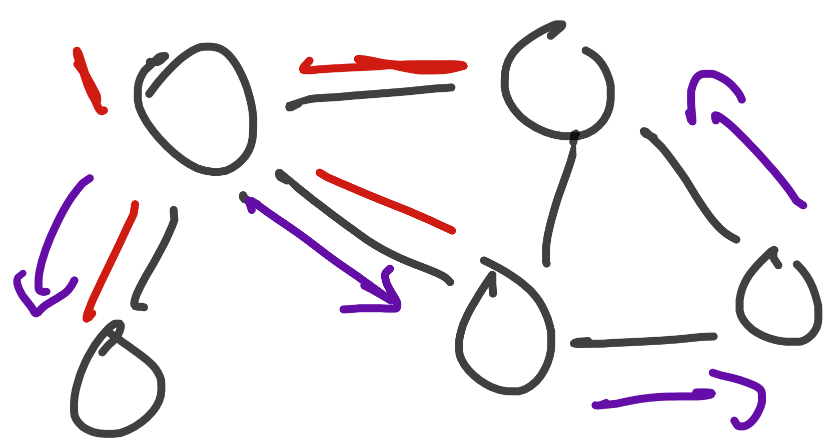
Approximating Y_u with a small random sample $W \subseteq V$

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

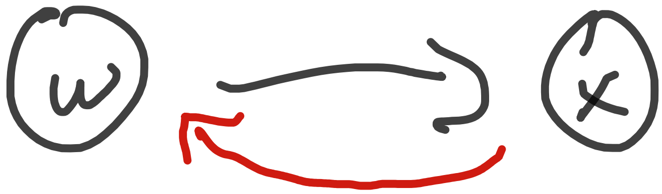
DFS - captures more macro-scale network properties

closer to role discovery?

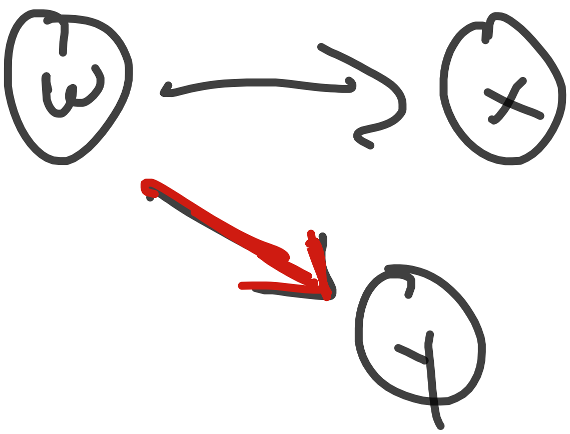
Actual strategy

Start: $w \rightarrow x$

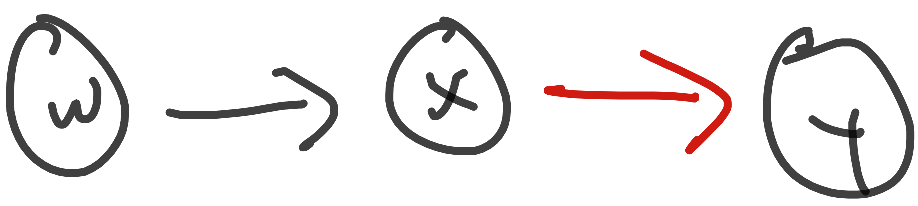
with prob. p, return to w



with prob. q, BFS



with prob. $1-p-q$, DFS



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