Talkin' proj

- Proposals due in a couple weeks
- Order will be random
- Keep an eye/ear open for more specific details

IMPORTANT → must use or consider a real-world or synthetic graph dataset and use some "graph mining"

(theory studies maybe ok, but talk to me)

Some topics:

* studying real-world network properties, especially for novel data
* prediction → predicting links, metadata, edge direction

* classification → determining properties of a vertex, groups or communities

* clustering → technique for classification, finding communities or similar structures

* diffusion or simulation → how "information", data, etc. propagates through a network

* data creation → creating a novel graph dataset

Link Prediction

Basically: inferring the growth process of network → predicting which links are most likely to form

Facebook: who you should add as friends

Amazon: what you will buy

For the above:

Recommender Systems

Also: inferring "hidden" links

FB: friends in reality, but not in the graph

- Impropiety
- Terrorists "hiding in plain sight"

Issue: intentional noise

Other examples: financial transaction network

→ Problems reduce to eliminating noise and finding the most probable actual structure

---

Our general approach for LP
link prediction

* Consider network at time t_0
* Use topology and/or metadata to predict most probable links
* Validate our guesses at $t_0 + \Delta t$
  → can compare various methods

OR

OR

use train+test (+validation)

{ on a single graph snapshot

→ useful for the "hidden links"

Unsupervised methods for LP

→ not explicity training on same ground truth

For LP: we use observed emperical growth processes

→ triadic closure
preferential attachment

Our methods

* Common neighbors: $C(x,y) = |N(x) \cap N(y)|$
* preferential attachment: $P(x,y) = |N(x)| * |N(y)|$
* Jaccard Index: $J(x,y) = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|}$
* Adamic-Adar: $A(x,y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{\log(|N(u)|)}$

→ considering common neighbors

↳ considering common neighbors of x,y and biasing against large degree

Why bias against high degree?

Some networks have high degree assortativity

↳ low degrees tend to attach to low degrees and high to high

* Katz: $K(x,y) = \sum_{l=1}^{\infty} \beta^l \left| paths_{x,y}^{<l>} \right|$

l = length of path

β = weighting of importance

small β → Katz becomes similar to common neighbors

$\left| paths_{x,y}^{<l>} \right|$ = # of paths from x to y of length l

↳ can get this via BFS

$$K = (I - \beta A)^{-1} - I$$

A: adjacency matrix; I: identity matrix

* Personalized Pagerank: what's the probability of a random walk from

* Personalized Pagerank: what's the probability on a random walk from x the walk will reach y

↳ we randomly traverse from adjacencies

(More next week)

---

* Code mode *

- consider the ONC temporal network
- compare some of the above as we already did with common neighbors and triadic closure

Here: we're just considering topological properties at some t_0 with no real notion of a ground truth

↳ unsupervised

Results: common neighbors did

★ fantastic ★

Other one were "alright"

↳ we already observed triadic closure properties in the network, which make for good predictions

closure properties
which make for good predictions

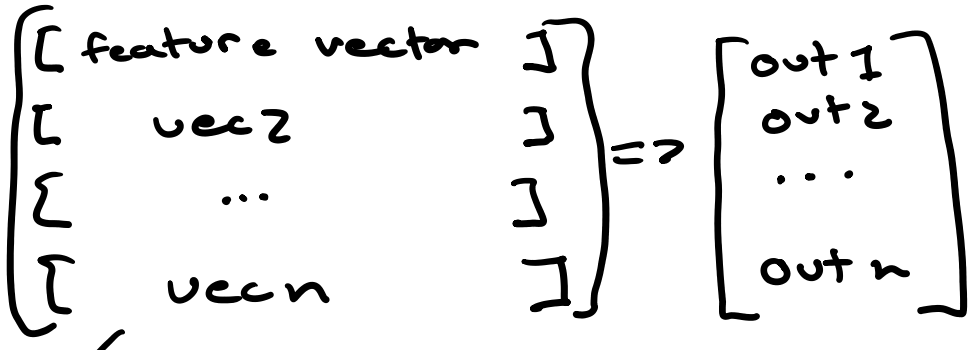
For HW 2:

Other techniques like PR, AA

Other network types that don't follow triadic closure

Learning on graphs (light intro)

Challenge: traditional supervised ML problems



For graphs → how do we capture topological properties?

→ this is pretty hard to turn into an explict set of features

SO: Representative learning

So: Representative Learning

(we construct features using same procedure that avoids hand-selecting feature properties)

Instead, we construct "embeddings"

Matrix Factorization:

features interact

$A = UVU^T$

adjacency

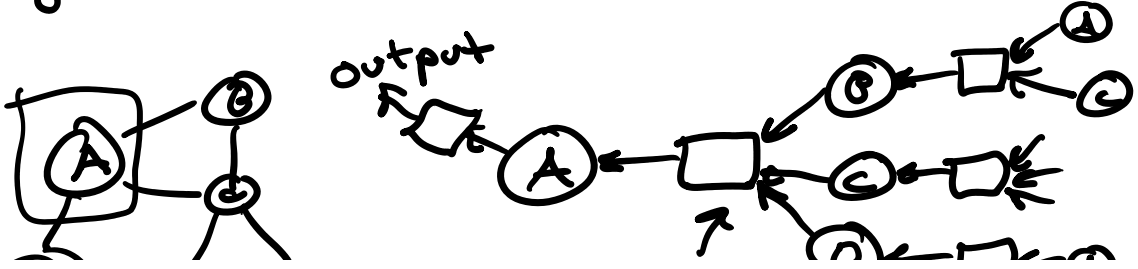
latent feature

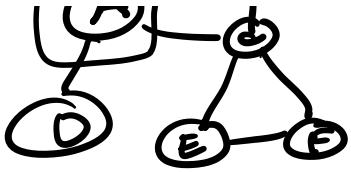
these arise via solving some optimization problem

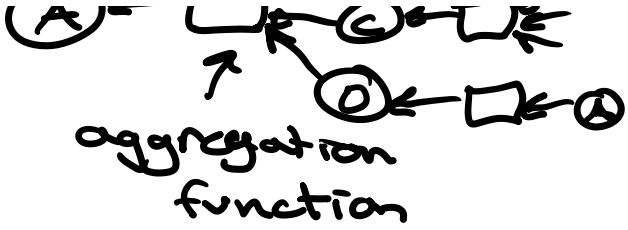
Random Walks:

Capturing statistical information that arises from random walks

Neighborhood based encoders:




D
E
F


aggregation
function