Lecture 8 - Link Prediction

Thursday, February 12, 2026 9:23 AM

Plan o' today:

- HW1 discussion (Note: you still need to calculate Gini)
- Proposal discussion
- Lit. review discussion
- Matthew presents: The Link Prediction Problem for Social Networks
- Link prediction
- General learnin' on graphs

HW1: Yes calculate

↳ B a faster way

FWBW: can make some modifications

I.e., can have set of vertices to avoid

(by computing subgraphs)

Proposal:

10-15 min UG

10+15 for G

(proposal) (lit. review)

Submit presentation by Friday (Feb 20th)

Submit presentation by Friday (Feb 20th)

=> Follow document on the website (proposal.pdf)

Lit Review:

5 papers

↳ similar style to paper presentation, but less detail, more summarized

* Overall intro
  - Each paper, intro, background, results, discussion, etc.

* Overall conclusion, discussion

↳ Writeup: should be of the style you see for conference of journal

(Intro + Background (or prior work) if you were to write a paper on your topic)

Link Prediction

Basically: inferring the growth process of a network

↳ prediction of what edges are most likely

Facebook: who to add as friends

Amazon: what will you buy (edge from person → product)

↳ Recommender system (next class)

Also: inferring "hidden" edges

FB: friends in reality, not online

↳ Why?

- Impropriety
- "Malicious Actors"
  - ↳ Hiding in plain sight

Issue: Intentional noise

Eg, criminal organizations, [illegible]

eg., criminal organizations,
financial transaction networks

→ In general, problem reduces to eliminate noise and finding the most probable topology

---

Our general approach for LP

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

OR

* Use a train + test (+validation) set on a single graph snapshot (for hidden links)

Unsupervised methods for LP

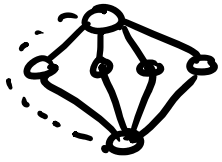
Unsupervised methods for LP

↳ not explicitly training on same ground truth

For LP: we use emperical growth processes

↳ triadic closure
pref. attachment

Our Methods (from paper)



* 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

$$A(x,y) = \sum \frac{1}{(\quad)}$$

$$A(x,y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{\log(|N(u)|)}$$

↳ comman neighbors, but scaling against large degree verts

Why: <u>assortitivity</u>

↳ vertices are more likely to connect to other vertices of a similar degree

* Katz

$$K(x,y) = \sum_{l=1}^{\infty} B^l |paths_{x,y}^{<l>}|$$

↑ # paths of length l between x, y

l = length of path

B = importance weight constant

↳ small B → comman neighbors

$|paths_{x,y}^{<l>}|$ → can get via BFS

$$K = (I - BA)^{-1} - I$$

↑ adj matrix ↖ identity matrix

* Personalized pagerank (random walk)

PPR(x,y) = prob. on a random walk from x we reach y

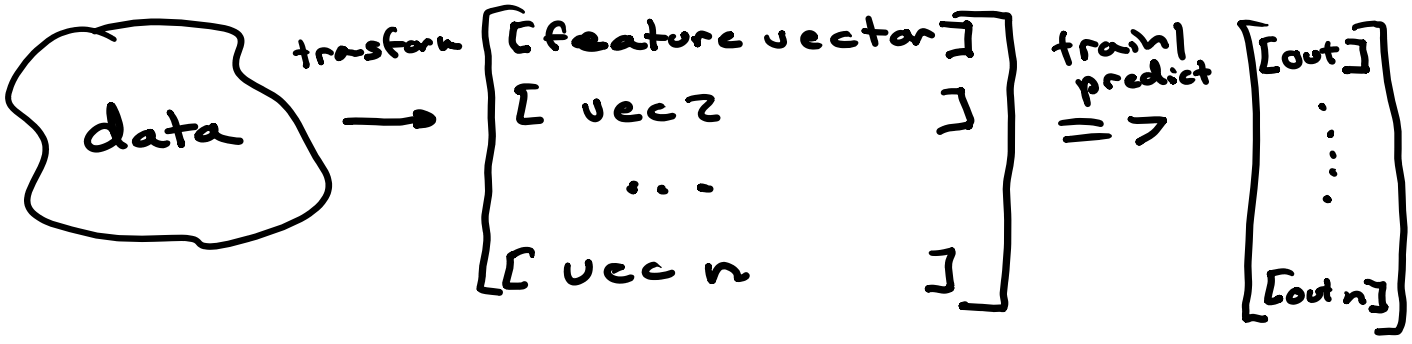
(prob distribution of all verts we reach)

(More later)

Learning on graphs (light intro)

(semi) supervised)

Challenge: traditional supervised ML problems



So the challenge:

How do we capture graph topology?

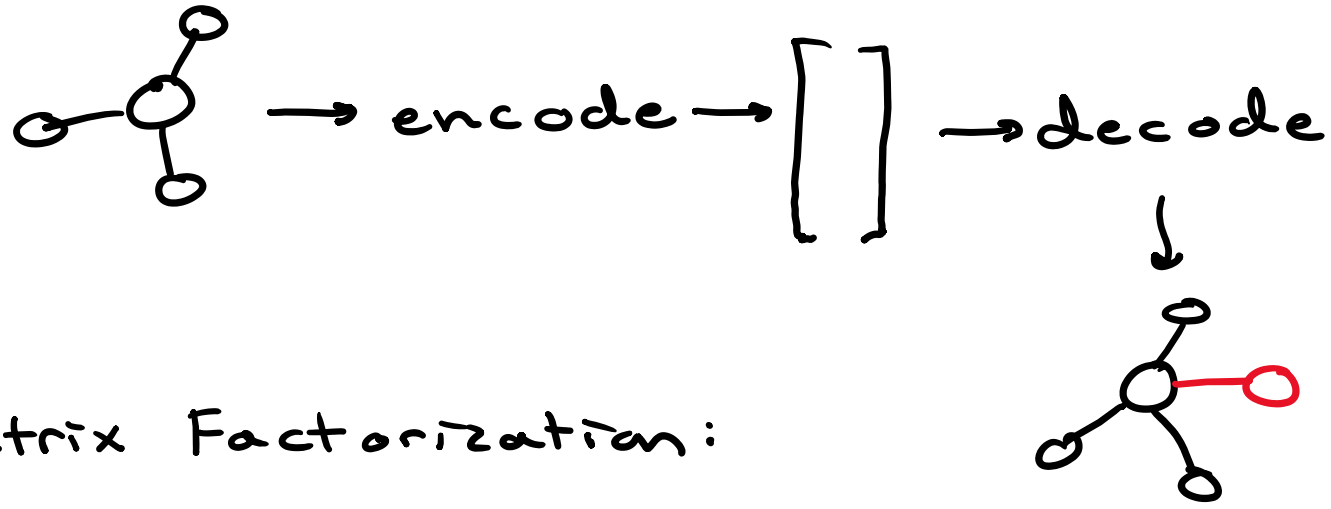
(

↳ pretty hard to turn into a fixed set of features

Representative Learning

We construct features using some procedure that avoids hand-selecting feature properties

↳ E.g. we construct "embeddings"



Matrix Factorization:

$$A = UVU^T$$

- A: adj matrix
- U: latent features
- V: feature interactions

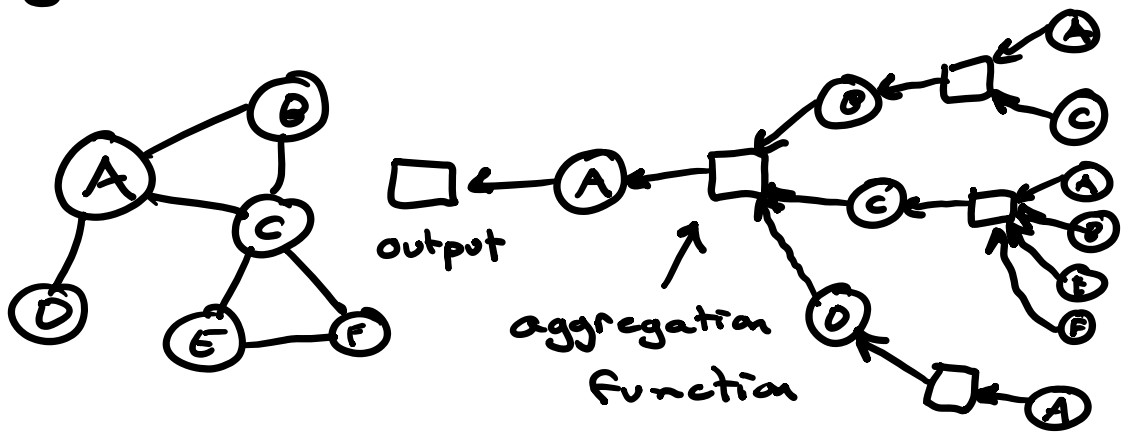
↳ not explicitly realized

* We solve an optimization

↳ not explicitly realized

* We solve an optimization problem by minimizing error between $A, (UVU^T)$

Neighborhood-based Encoders:



(GNNs)

Random Walks:

Capturing statistical information about the immediat neighborhood