Link Prediction Based on Graph Neural Networks

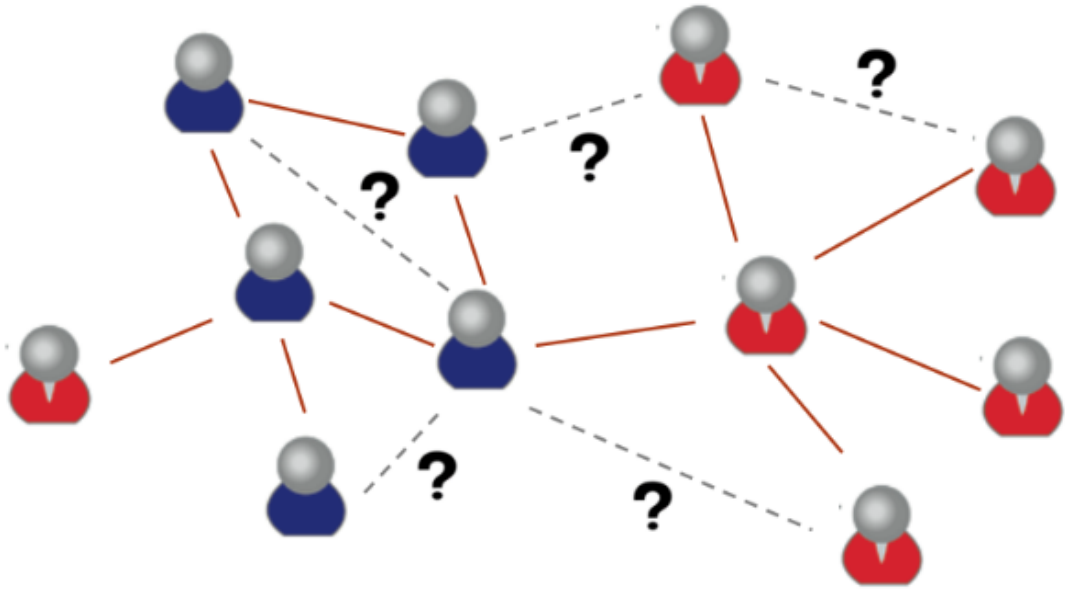
Zhang and Chen
NeurIPS 2018

Presenter: Jack Lanchantin
https://qdata.github.io/deep2Read

Outline

Outline

Link Prediction



- **Goal:** Predict whether two nodes in a network are likely to have a link
- Friend recommendation, movie recommendation, knowledge graph completion, metabolic network reconstruction, etc.

Outline

Heuristic Methods

- Compute heuristic node similarity scores as the likelihood of links
- Existing heuristics can be categorized based on the maximum hop of neighbors needed to calculate the score
- We define h-order heuristics to be those heuristics which require knowing up to h-hop neighborhood of the target nodes.

Heuristic Methods

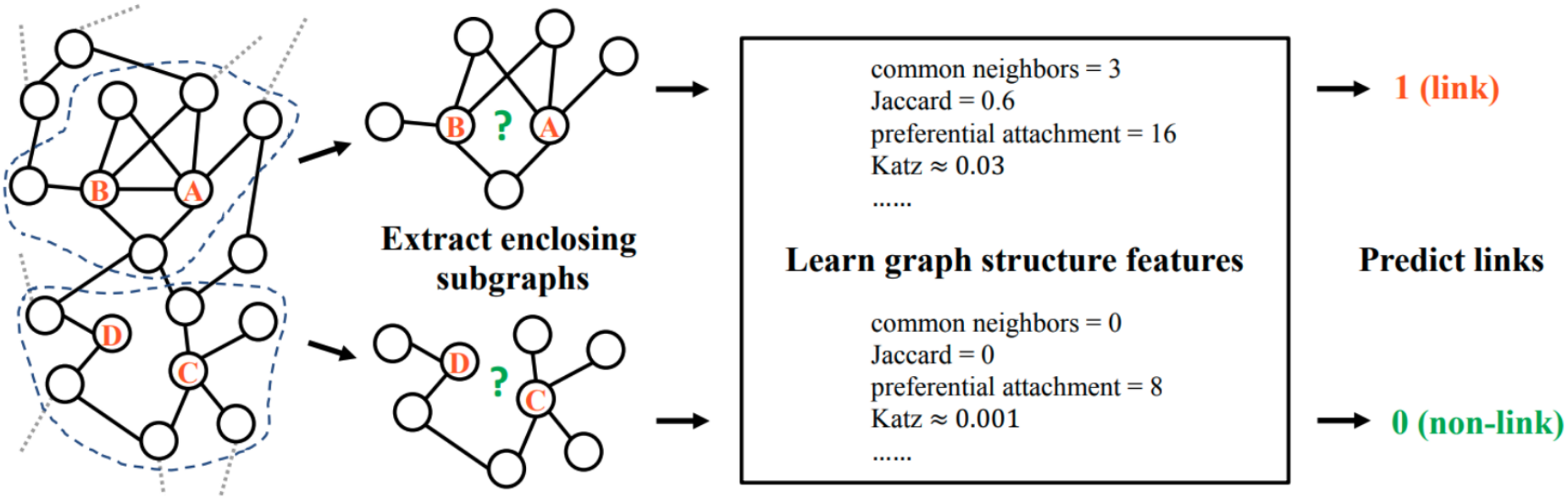
- Strong assumptions on when links may exist
- E.g. the common neighbor heuristic assumes that two nodes are more likely to connect if they have many common neighbors
  - This assumption may be correct in social networks, but is shown to fail in protein-protein interaction (PPI) networks - two proteins sharing many common neighbors are actually less likely to interact

Graph Structure Feature Methods

- Graph structure features are those features located inside the observed node and edge structures of the network, which can be calculated directly from the graph.
- Since heuristics can be viewed as predefined graph structure features, a natural idea is to automatically learn such features from the network

Link Prediction using Graph Structure Feature Methods

Zhang et. al. 2017 extract local enclosing subgraphs around links as the training data, and use a fully-connected neural network to learn which enclosing subgraphs correspond to link existence



High-order Heuristics

- However, it is shown that high-order heuristics often have much better performance than first and second-order ones
- To learn good high-order features, it seems we need a large hop number h so that the enclosing subgraph becomes the entire network
  - Unaffordable time and memory for most practical networks
- But do we really need such a large h to learn high-order heuristics?

Contributions of This Paper

1. Present a new theory for learning link prediction heuristics, justifying learning from **local** subgraphs instead of entire networks

Contributions of This Paper

1. Present a new theory for learning link prediction heuristics, justifying learning from **local** subgraphs instead of entire networks
2. Propose SEAL, a novel link prediction framework based on GNN, outperforming previous methods

Outline

Notations

- Let $G = (V, E)$ be an undirected graph, where V is the set of vertices, E is the set of observed links, and A is the adjacency matrix

A theory for unifying link prediction heuristics

- Aim to understand deeper the mechanisms behind various link prediction heuristics, motivating the idea of learning heuristics from local subgraphs

Enclosing Subgraphs

Definition

(Enclosing subgraph) For a graph $G = (V, E)$, given two nodes $x, y \in V$, the h-hop enclosing subgraph for (x, y) is the subgraph $G^h_{x,y}$ induced from G by the set of nodes $\{ i \mid d(i, x) \leq h \text{ or } d(i, y) \leq h \}$.

- The enclosing subgraph describes the "h-hop surrounding environment" of (x, y)

Theorem

Any h-order heuristic for (x, y) can be accurately calculated from $G^h_{x,y}$.

- For example, a 2-hop enclosing subgraph will contain all the information needed to calculate any first and second-order heuristics

γ-decaying heuristic

Definition

(γ-decaying heuristic) A γ-decaying heuristic for (x, y) defined as:

$$\mathcal{H}(x, y) = \eta \sum_{l=1}^{\infty} \gamma^l f(x, y, l), \tag{1}$$

where γ is a decaying factor between 0 and 1, η is a positive constant, f is a nonnegative function of x, y, l under the the given network.

- We show that under certain conditions, a γ-decaying heuristic can be approximated from an h-hop enclosing subgraph, and the approximation error decreases at least exponentially with h

High-order Heuristics

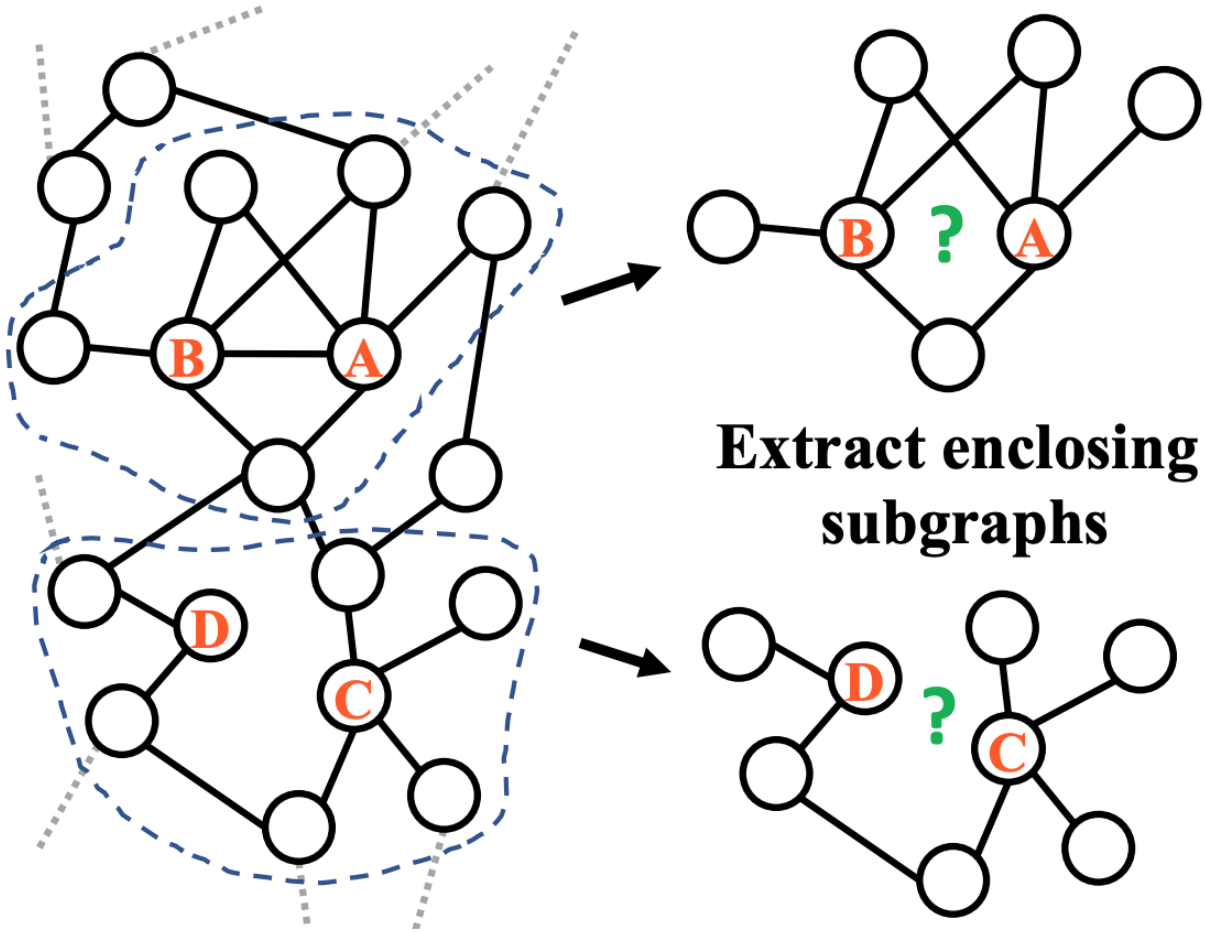
- Most high-order heuristics can be unified by a γ-*decaying theory*
- Since any γ-decaying heuristic can be approximated from an h-hop enclosing subgraph, and approximation error decreases at least exponentially with h: **we can safely use even a small h to learn good high-order features**
- It also implies that the "effective order" of these high-order heuristics is not that high

Outline

SEAL: GNN for Link Prediction

- SEAL does not restrict the learned features to be in some particular forms such as γ-decaying heuristics, but instead learns general graph structure features for link prediction.
- It contains three steps:
  1. subgraph extraction
  2. node information matrix construction
  3. GNN learning

Subgraph Extraction



Node Labeling

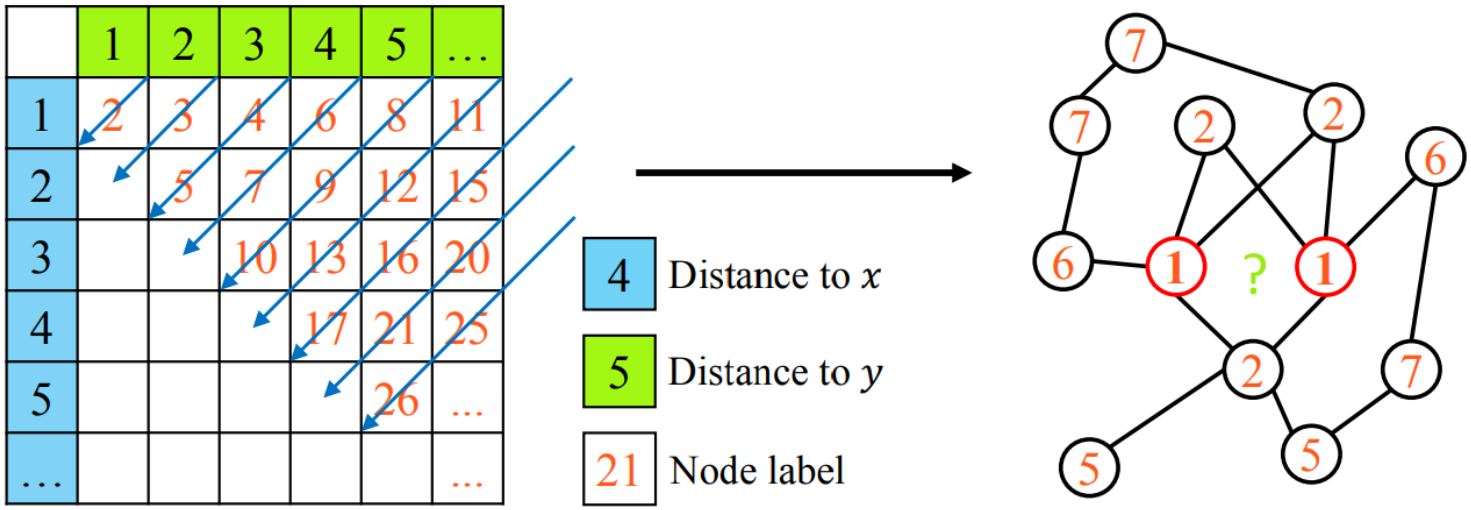
- The first component in X is each node's structural label
- Structural label: use different labels to **mark nodes' different roles** in an enclosing subgraph:
  1. The center nodes x and y are the target nodes between which the link is located
  2. Nodes with different relative positions to the center have different structural importance to the link

Structural Node Labeling Criteria

- The two target nodes x and y always have the distinctive label "1"
- Nodes i and j have the same label if $d(i, x) = d(j, x)$ and $d(i, y) = d(j, y)$

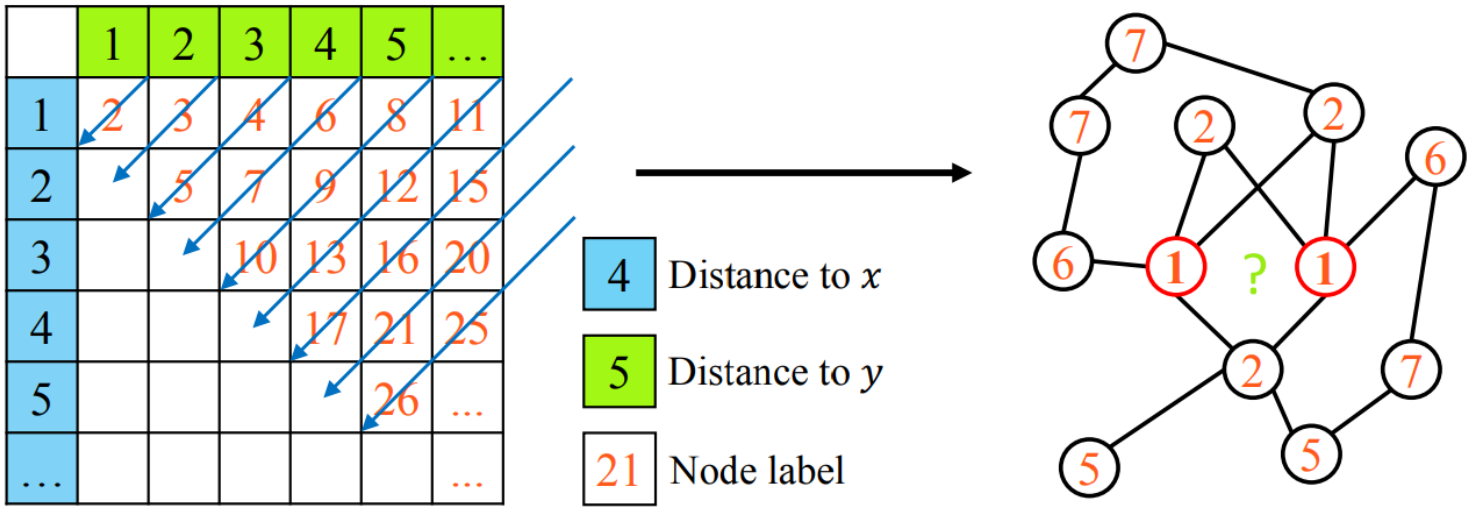
Double-Radius Node Labeling

For any node i with $(d(i,x), d(i,y)) = (1,1)$, assign label $f_l(i) = 2$. Nodes with radius $(1,2)$ or $(2,1)$ get label 3. Nodes with radius $(1,3)$ or $(3,1)$ get 4. Nodes with $(2,2)$ get 5, and so forth



Double-Radius Node Labeling

For any node i with $(d(i,x), d(i,y)) = (1,1)$, assign label $f_l(i) = 2$. Nodes with radius $(1,2)$ or $(2,1)$ get label 3. Nodes with radius $(1,3)$ or $(3,1)$ get 4. Nodes with $(2,2)$ get 5, and so forth



$$f_l(i) = 1 + min(d_x, d_y) + (d/2)[(d/2) + (d\%2) - 1], \quad (2)$$

where $d_x := d(i,x)$, $d_y := d(i,y)$, $d := d_x + d_y$, $(d/2)$ and $(d\%2)$ are the integer quotient and remainder of d divided by 2, respectively

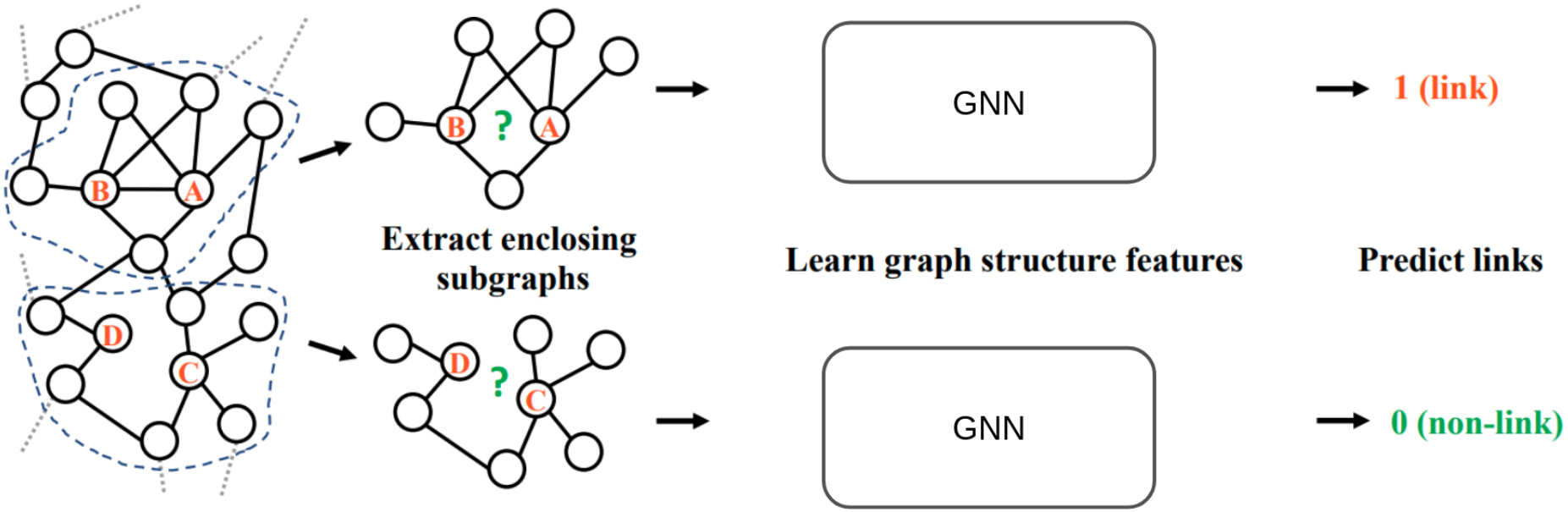
Incorporating latent and explicit features

- Other than the structural node labels, the node information matrix X also provides an opportunity to include latent and explicit features.
- By concatenating each node's embedding/attribute vector to its corresponding structural label, we can make SEAL simultaneously learn from all three types of features

GNN Learning

- Given a subgraph and the node embeddings, learn GNN to predict link/no link
- Use the Deep Graph Convolutional Neural Network from Zhang et. al AAAI 2018

SEAL (learning from Subgraphs, Embeddings and Attributes for Link prediction)



Outline

Datasets

- 8 datasets: US Air lines, collaboration network of researchers, protein-protein interaction, electrical grid, router-level Internet, C. elegans, reaction network of metabolites in E. coli, network of US political blogs
- Randomly remove 10% existing links from each dataset as positive testing data.
- Randomly sample nonexistent links (unconnected node pairs) as negative testing data

Comparison to Heuristic Methods

- First compare SEAL with methods that only use graph structure features (i.e. no unsupervised feature learning such as DeepWalk embeddings)
- Select h only from $\{1, 2\}$ to validate our theoretical results that the most useful information is within local structures (also empirically observed accuracy didn't increase for $h > 2$)

Comparison with heuristic methods (AUC)

Data	CN	Jaccard	PA	AA	RA	Katz	PR	SR	ENS	WLK	WLNM	**SEAL**
USAir	93.80±1.22	89.79±1.61	88.84±1.45	95.06±1.03	95.77±0.92	92.88±1.42	94.67±1.08	78.89±2.31	88.96±1.44	**96.63**±0.73	95.95±1.10	**96.62**±0.72
NS	94.42±0.95	94.43±0.93	68.65±2.03	94.45±0.93	94.45±0.93	94.85±1.10	94.89±1.08	94.79±1.08	97.64±0.25	98.57±0.51	98.61±0.49	**98.85**±0.47
PB	92.04±0.35	87.41±0.39	90.14±0.45	92.36±0.34	92.46±0.37	92.92±0.35	93.54±0.41	77.08±0.80	90.15±0.45	93.83±0.59	93.49±0.47	**94.72**±0.46
Yeast	89.37±0.61	89.32±0.60	82.20±1.02	89.43±0.62	89.45±0.62	92.24±0.61	92.76±0.55	91.49±0.57	82.36±1.02	95.86±0.54	95.62±0.52	**97.91**±0.52
C.ele	85.13±1.61	80.19±1.64	74.79±2.04	86.95±1.40	87.49±1.41	86.34±1.89	**90.32**±1.49	77.07±2.00	74.94±2.04	89.72±1.67	86.18±1.72	**90.30**±1.35
Power	58.80±0.88	58.79±0.88	44.33±1.02	58.79±0.88	58.79±0.88	65.39±1.59	66.00±1.59	76.15±1.06	79.52±1.78	82.41±3.43	84.76±0.98	**87.61**±1.57
Router	56.43±0.52	56.40±0.52	47.58±1.47	56.43±0.51	56.43±0.51	38.62±1.35	38.76±1.39	37.40±1.27	47.58±1.48	87.42±2.08	94.41±0.88	**96.38**±1.45
E.coli	93.71±0.39	81.31±0.61	91.82±0.58	95.36±0.34	95.95±0.35	93.50±0.44	95.57±0.44	62.49±1.43	91.89±0.58	96.94±0.29	97.21±0.27	**97.64**±0.22

Outline

Conclusions

- This paper presented:
  1. Theoretical justifications for learning link prediction heuristics from local enclosing subgraphs
  2. GNN on local subgraphs
- Cons: didn't show examples/reasons where higher-order methods might perform better

Popular heuristics for link prediction

Name	Formula	Order
common neighbors	$\lvert\Gamma(x) \cap \Gamma(y)\rvert$	first
Jaccard	$\frac{\lvert\Gamma(x)\cap\Gamma(y)\rvert}{\lvert\Gamma(x)\cup\Gamma(y)\rvert}$	first
preferential attachment	$\lvert\Gamma(x)\rvert \cdot \lvert\Gamma(y)\rvert$	first
Adamic-Adar	$\sum_{z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\log\lvert\Gamma(z)\rvert}$	second
resource allocation	$\sum_{z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\lvert\Gamma(z)\rvert}$	second
Katz	$\sum_{l=1}^{\infty} \beta^l \lvert\text{walks}^{\langle l\rangle}(x,y)\rvert$	high
PageRank	$[\pi_x]_y + [\pi_y]_x$	high
SimRank	$\gamma \frac{\sum_{a\in\Gamma(x)}\sum_{b\in\Gamma(y)}\text{score}(a,b)}{\lvert\Gamma(x)\rvert\cdot\lvert\Gamma(y)\rvert}$	high

Notes: $\Gamma(x)$ denotes the neighbor set of vertex x. $\beta < 1$ is a damping factor. $\lvert\text{walks}^{\langle l\rangle}(x,y)\rvert$ counts the number of length-l walks between x and y. $[\pi_x]_y$ is the stationary distribution probability of y under the random walk from x with restart, see [9]. SimRank score is a recursive definition.

Comparison of different link prediction methods

	Heuristics	Latent features	WLK	WLNM	SEAL
Graph structure features	Yes	No	Yes	Yes	Yes
Learn from full h-hop	No	n/a	Yes	No	Yes
Latent/explicit features	No	Yes	No	No	Yes
Model	n/a	LR/inner product	SVM	NN	GNN