Neural Common Neighbor with Completion for Link Prediction

ICLR 2024

Xiyuan Wang, Haotong Yang, Muhan Zhang

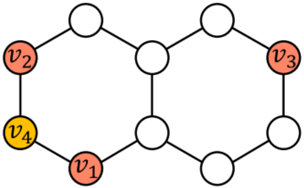
Institute of Artificial Intelligence, Peking University

April 25, 2024

MPNN for Link Prediction

Xiyuan Wang,
Haotong Yang,
Muhan Zhang

Vanilla MPNN fails in this task



- Learns node representation only.
- Cannot distinguish (v_1, v_2) and (v_1, v_3).

Structural feature like number of common neighbor can help.

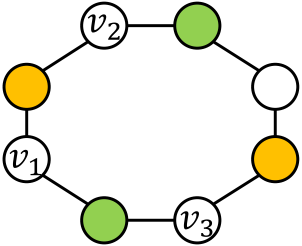
Structure Feature cannot Capture Node Feature

Xiyuan Wang,
Haotong Yang,
Muhan Zhang

Commonly structure features

$$\begin{array}{ll} \text{Common Neighbor} & \sum_{u \in N(i) \cap N(j)} 1 \\ \text{Resource Allocation} & \sum_{u \in N(i) \cap N(j)} \frac{1}{d(u)} \\ \text{Adamic Adar} & \sum_{u \in N(i) \cap N(j)} \frac{1}{\log d(u)} \end{array} \tag{1}$$

- unlearnable
- unable to capture node feature



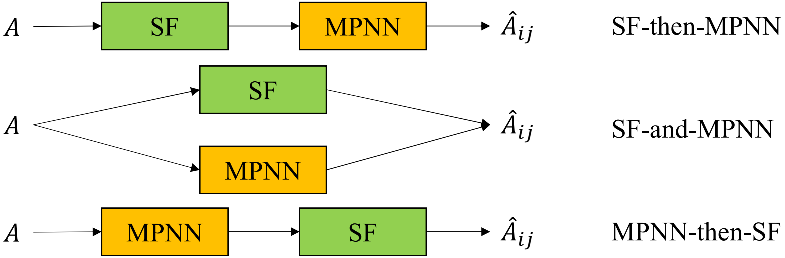
Cannot distinguish (v_1, v_2) and (v_1, v_3).

New Architecture

Xiyuan Wang,
Haotong Yang,
Muhan Zhang

Structural features (SF) like common neighbor are widely used.

- Existing works utilize SF in two ways.
  - SF-then-MPNN. Take SF as MPNN's input
    - Low scalability, need to rerun MPNN as the SF changes with target link.
  - SF-and-MPNN. Ensemble MPNN with SF.
    - MPNN and SF are completely separated. Low expressivity.
- We use SF to guide the pooling of MPNN's output (MPNN-then-SF).
  - Good scalability and expressivity.



Connection to Previous Work

Xiyuan Wang, Haotong Yang, Muhan Zhang

$$\left.\begin{array}{ll}\text{Common Neighbor} & \sum_{u\in N(i)\cap N(j)} 1\\ \text{Resource Allocation} & \sum_{u\in N(i)\cap N(j)} \frac{1}{d(u)}\\ \text{Adamic Adar} & \sum_{u\in N(i)\cap N(j)} \frac{1}{\log d(u)}\\ \text{Neo-GNN} & \sum_{u\in N_1^{l_1}(i)\cap N_1^{l_2}(j)} A_{iu}^{l_1}A_{ju}^{l_2} f(d(u))\\ \text{BUDDY} & \sum_{u\in N_{l_1}^1(i)\cap N_{l_2}^1(j)} 1, \sum_{u\in N_l^1(i)-\bigcup_{l'=1}^k N_{l'}^1(j)} 1\end{array}\right\}\Rightarrow \tag{2}$$

$$\Rightarrow \sum_{u\in N_{l_1}^{l_2}(i)\oplus N_{l_1'}^{l_2'}(j)} g(A_{iu}^{l_2})g(A_{ju}^{l_2'})\mathrm{MPNN}(u,A,X) \tag{3}$$

Higher-order neighbors and neighborhood differences lead to negligible performance gain, leading to our NCN model:

$$\mathrm{NCN}(i,j,A,X) = \sum_{u\in N(i)\cap N(j)} \mathrm{MPNN}(u,A,X) \tag{4}$$

Incompleteness Visualization

Xiyuan Wang, Haotong Yang, Muhan Zhang

Incompleteness of graph is ubiquitous in link prediction tasks.

- The target edge exists in input graph on training set but not on test set.

Besides the target edge, other edges, like those connected to common neighbors, is also affected.

To visualize it, we assume that

- Graph with training set edges only is the *incomplete* graph.
- Graph with training, validation and test set edges is the *complete* graph.

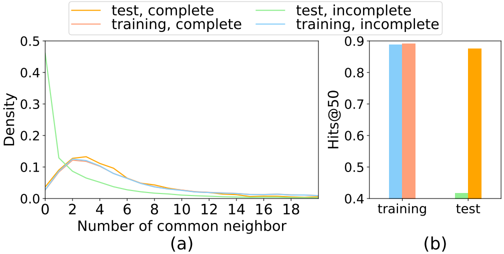
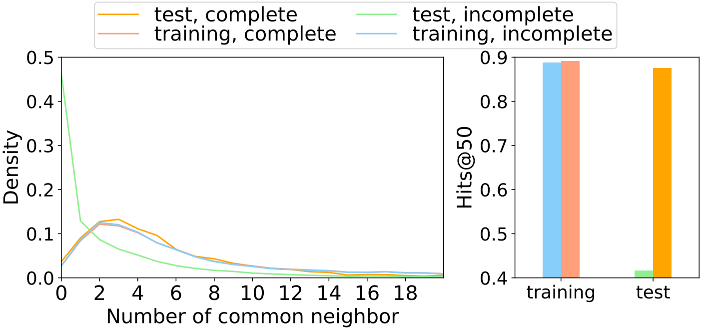


Figure: Ogbl-collab dataset (a) distribution of common neighbor (b) performance of common neighbor

Incompleteness Visualization: Distribution Shift

Xiyuan Wang,
Haotong Yang,
Muhan Zhang



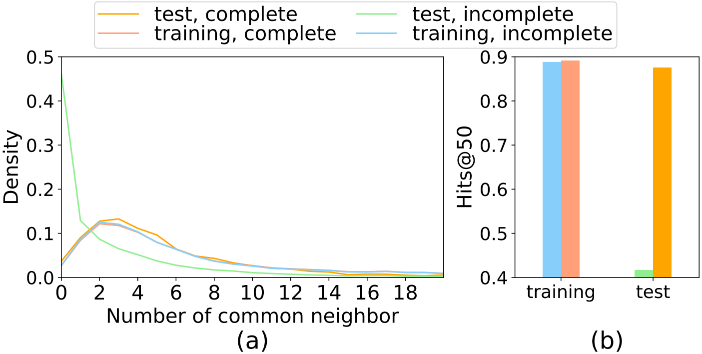
(a) (b)

- blue and green lines in (a): a significant *distribution shift* between the training and test sets in the incomplete graph of the ogbl-collab dataset.
- red and orange lines: shift disappears when the graph is complete.

Distribution shifts can enlarge the gap between training and test error.

Incompleteness Visualization: Loss of Common Neighbor

Xiyuan Wang,
Haotong Yang,
Muhan Zhang

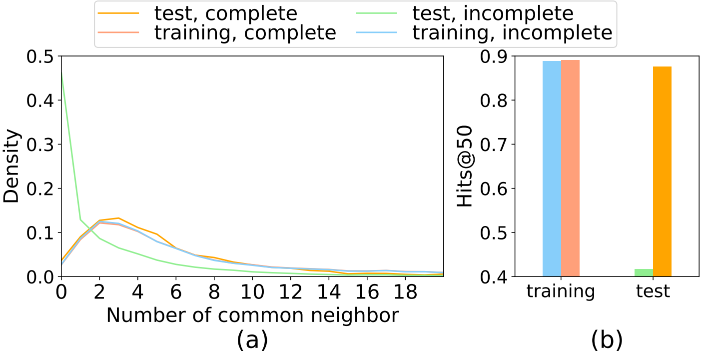


- Blue and green lines in (a): there are fewer common neighbors in the incomplete graph.

Loss of Common Neighbor Information can lead to high training error and thus high test error.

Incompleteness Visualization: Performance Degradation

Xiyuan Wang,
Haotong Yang,
Muhan Zhang



- (b): Imcompleteness of non-target links leads to significant performance degradation.

Common Neighbor Completion

Due to imcompleteness, NCN can use not only the common neighbors in the input graph. It can also predict unobserved common neighbors.
Given a target link (i, j), the probability that u is a common neighbor of (i, j) is

$$P_{uij} = \begin{cases} 1 & \text{if } u \in N(i,A) \cap N(j,A) \\ \sigma(\mathrm{NCN}(i, u, A, X)) & \text{if } u \in N(j,A) - N(i,A) \\ \sigma(\mathrm{NCN}(j, u, A, X)) & \text{if } u \in N(i,A) - N(j,A) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

NCN with Completion (NCNC) becomes

$$\mathrm{NCNC}(i, j, A, X) = \sum_{u \in V} P_{uij} \mathrm{MPNN}(u, A, X). \tag{6}$$

Link Prediction

Xiyuan Wang, Haotong Yang, Muhan Zhang

	Cora	Citeseer	Pubmed	Collab	PPA	Citation2	DDI
Metric	HR@100	HR@100	HR@100	HR@50	HR@100	MRR	HR@20
CN	33.92±0.46	29.79±0.90	23.13±0.15	56.44±0.00	27.65±0.00	51.47±0.00	17.73±0.00
AA	39.85±1.34	35.19±1.33	27.38±0.11	64.35±0.00	32.45±0.00	51.89±0.00	18.61±0.00
RA	41.07±0.48	33.56±0.17	27.03±0.35	64.00±0.00	49.33±0.00	51.98±0.00	27.60±0.00
GCN	66.79±1.65	67.08±2.94	53.02±1.39	44.75±1.07	18.67±1.32	84.74±0.21	37.07±5.07
SAGE	55.02±4.03	57.01±3.74	39.66±0.72	48.10±0.81	16.55±2.40	82.60±0.36	53.90±4.74
SEAL	81.71±1.30	83.89±2.15	75.54±1.32	64.74±0.43	48.80±3.16	87.67±0.32	30.56±3.86
NBFnet	71.65±2.27	74.07±1.75	58.73±1.99	OOM	OOM	OOM	4.00±0.58
NeoGNN	80.42±1.31	84.67±2.16	73.93±1.19	57.52±0.37	49.13±0.60	87.26±0.84	63.57±3.52
BUDDY	88.00±0.44	<u>92.93±0.27</u>	74.10±0.78	<u>65.94±0.58</u>	49.85±0.20	87.56±0.11	78.51±1.36
NCN	<u>89.05±0.96</u>	91.56±1.43	<u>79.05±1.16</u>	64.76±0.87	<u>61.19±0.85</u>	<u>88.09±0.06</u>	<u>82.32±6.10</u>
NCNC	**89.65±1.36**	**93.47±0.95**	**81.29±0.95**	**66.61±0.71**	**61.42±0.73**	**89.12±0.40**	**84.11±3.67**

Ablation Study

Xiyuan Wang, Haotong Yang, Muhan Zhang

	Cora	Citeseer	Pubmed	Collab	PPA	Citation2	DDI
Metric	HR@100	HR@100	HR@100	HR@50	HR@100	MRR	HR@20
CN	33.92±0.46	29.79±0.90	23.13±0.15	56.44±0.00	27.65±0.00	51.47±0.00	17.73±0.00
GAE	89.01±1.32	91.78±0.94	78.81±1.64	36.96±0.95	19.49±0.75	79.95±0.09	61.53±9.59
GAE+CN	88.61±1.31	91.75±0.98	79.04±0.83	64.47±0.14	51.83±0.58	87.81±0.06	80.71±5.56
NCN2	88.87±1.34	91.36±1.02	80.21±0.78	65.43±0.46	OOM	OOM	OOM
NCN-diff	89.12±1.04	91.96±1.23	80.28±0.88	64.08±0.40	57.86±1.26	86.68±0.16	17.67±8.70
NCN	89.05±0.96	91.56±1.43	79.05±1.16	64.76±0.87	61.19±0.85	88.09±0.06	82.32±6.10
NoTLR	85.46±1.65	88.08±1.23	76.59±1.33	64.22±0.49	60.66±0.63	88.64±0.14	66.52±11.37
NCNC	89.65±1.36	93.47±0.95	81.29±0.95	66.61±0.71	61.42±0.73	89.12±0.40	84.11±3.67
NCNC-2	89.14±0.84	93.14±0.96	81.41±1.07	66.80±0.43	$> 24h$	$> 24h$	$> 24h$

Scalability Comparison

Xiyuan Wang, Haotong Yang, Muhan Zhang

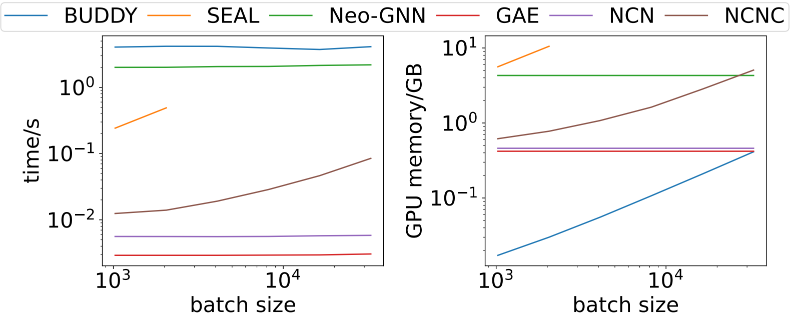


Figure: Inference time and GPU memory on ogbl-collab. The process we measure includes preprocessing and predicting one batch of test links. Relation between time y and batch size t is $y = B + Ct$, where B, C are model specific constants. SEAL has out-of-memory problem and only uses small batch sizes.