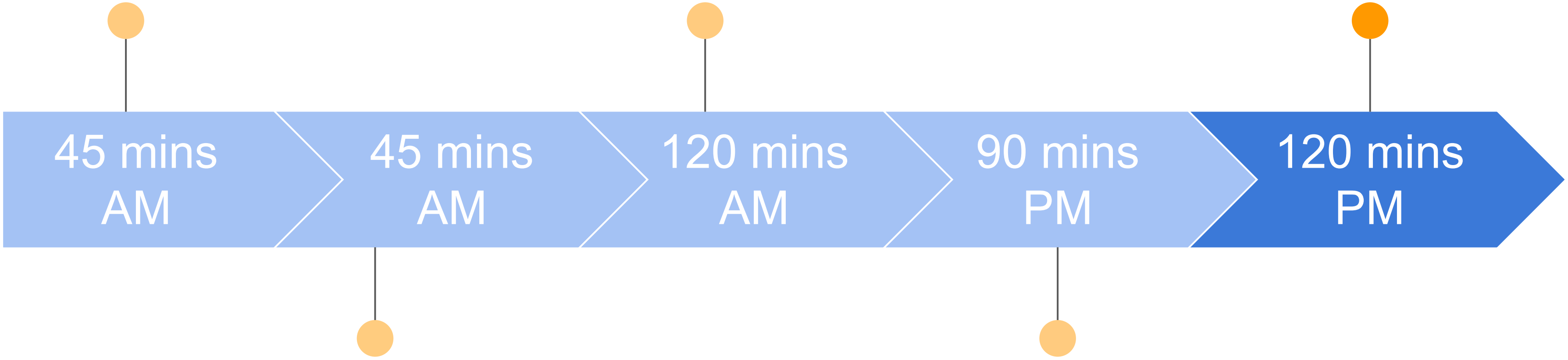
Learning From Networks

——*Algorithms, Theory, & Applications*

Xiao Huang, Peng Cui, Yuxiao Dong, Jundong Li, Huan Liu, Jian Pei, Le Song,
Jie Tang, Fei Wang, Hongxia Yang, Wenwu Zhu

xhuang@tamu.edu; cuip@tsinghua.edu.cn; yuxdong@microsoft.com; jundongl@asu.edu;
huan.liu@asu.edu; jpei@cs.sfu.ca; le.song@antfin.com; jietang@tsinghua.edu.cn;
few2001@med.cornell.edu; yang.yhx@alibaba-inc.com; wwzhu@tsinghua.edu.cn;



Motivations

45 mins
AM

Feature
Selection

45 mins
AM

Network
Embedding

120 mins
AM

Attributed Network
Embedding

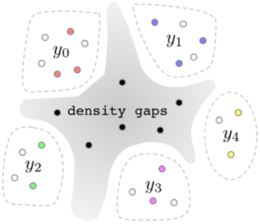
90 mins
PM

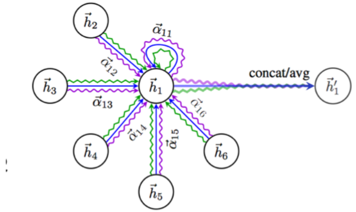
Graph Neural
Networks

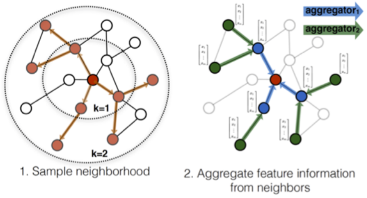
120 mins
PM

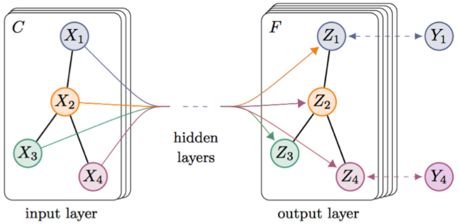
Graph neural networks

Graph Isomorphism Network, Deep Graph Infomax	2019: Velickovic et al. & Xu et al., ICLR’19
Graph attention	2018: Velickovic et al., ICLR’18
Neural message passing, GraphSage	2017: Gilmer et al., ICML’17; Hamilton et al., NIPS’17
Gated graph neural network	2016: Li et al., ICLR’16
structure2vec	2016: Dai et al., ICML’16
Graph convolutional network	2015: Duvenaud et al., NIPS’15; Kipf & Welling ICLR’17
Spectral graph convolution	2014: Bruna et al., ICLR’14
Graph neural network	2005: Gori et al., IJCNN’05

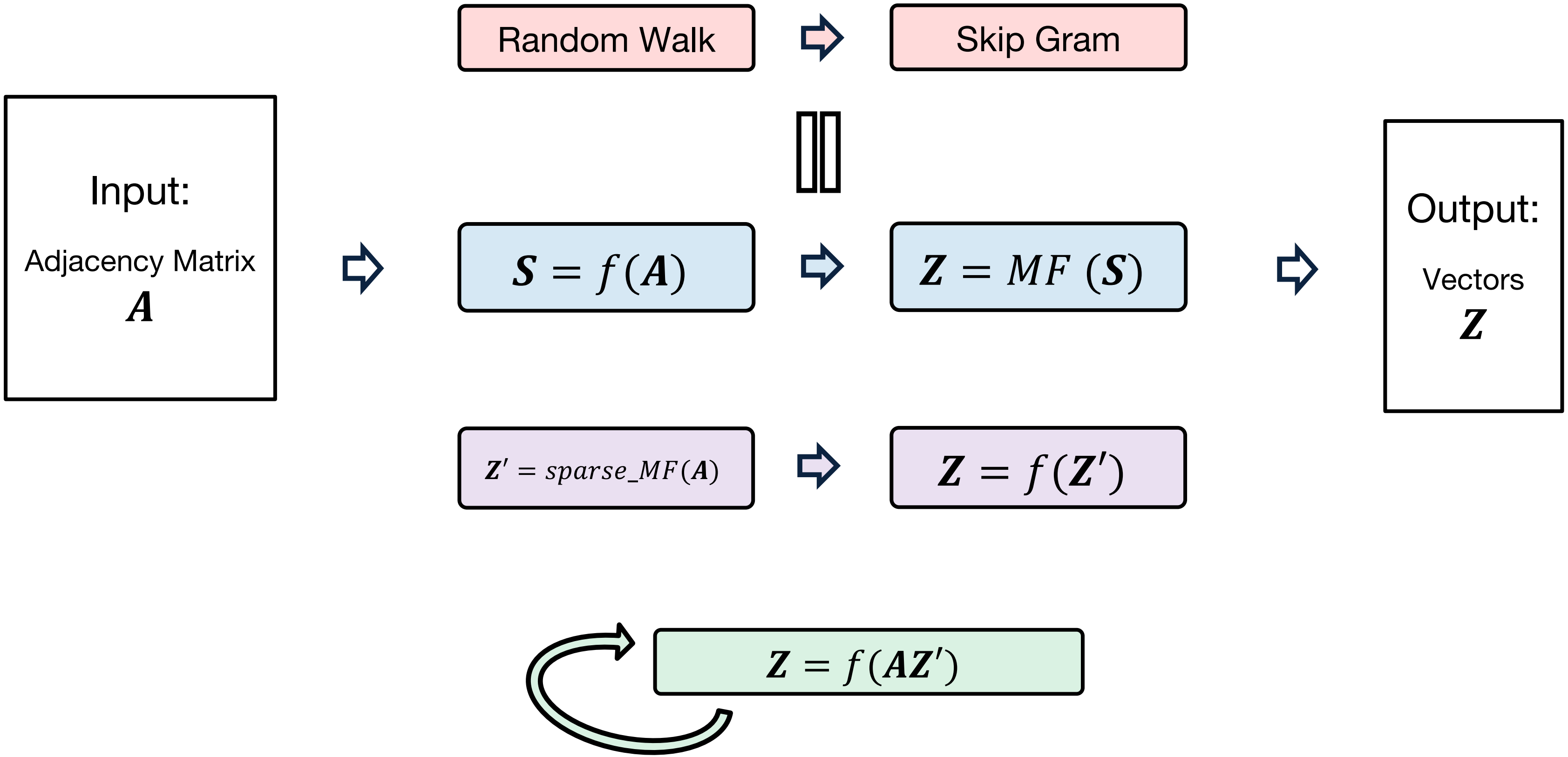








Connecting NE with graph neural networks



Input:

Adjacency Matrix

$\boldsymbol{A}$

Random Walk

Skip Gram

$\boldsymbol{S} = f(\boldsymbol{A})$

$\boldsymbol{Z} = MF\ (\boldsymbol{S})$

$\boldsymbol{Z}' = sparse\_MF(\boldsymbol{A})$

$\boldsymbol{Z} = f(\boldsymbol{Z}')$

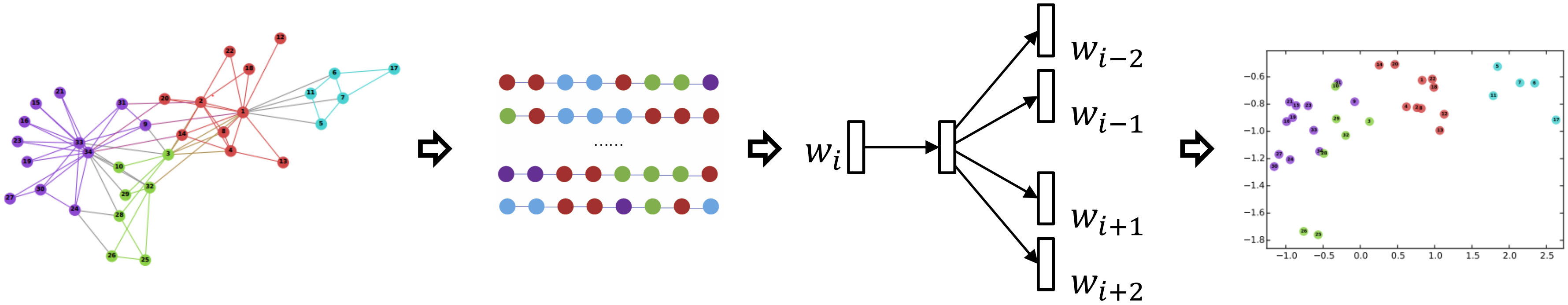
Output:

Vectors

$\boldsymbol{Z}$

$\boldsymbol{Z} = f(\boldsymbol{A}\boldsymbol{Z}')$

Network embedding: DeepWalk



Perozzi et al. DeepWalk: Online learning of social representations. In *KDD’ 14*, pp. 701–710. **Most Cited Paper in KDD’14.**

Random walk strategies

- Random Walk
    - DeepWalk (walk length > 1)
    - LINE (walk length = 1)
- Biased Random Walk
    - node2vec (2-order random walk)
    - metapath2vec (heterogeneous random walk)

1. Perozzi et al. **DeepWalk**: Online learning of social representations. In *KDD' 14*. **Most Cited Paper in KDD'14.**
2. Tang et al. **LINE**: Large scale information network embedding. In *WWW'15*. **Most Cited Paper in WWW'15.**
3. Grover and Leskovec. **node2vec**: Scalable feature learning for networks. *In KDD'16.* **2nd Most Cited Paper in KDD'16.**
4. Dong et al. **metapath2vec**: scalable representation learning for heterogeneous networks. In *KDD* 2017. **Most Cited Paper in KDD'17.**

Application: Embedding Heterogeneous Academic Graph

225,572,477
Papers

244,499,947
Authors

664,891
Topics

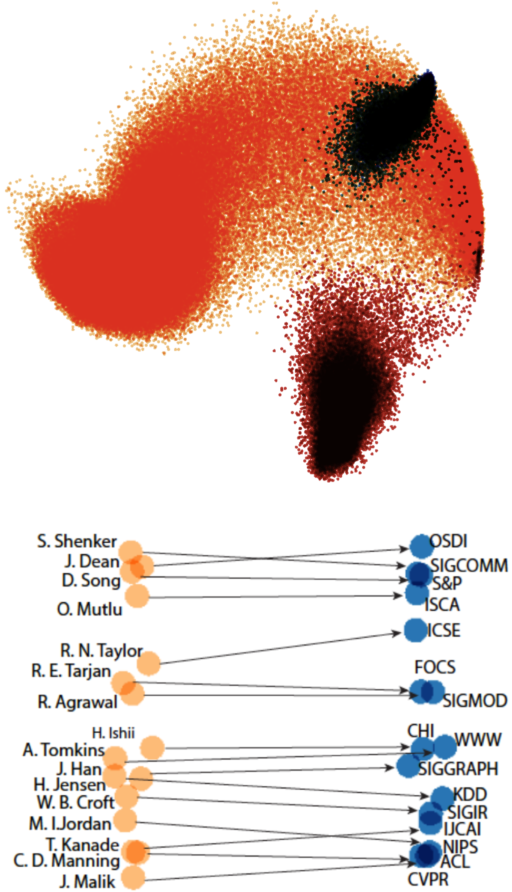
4,397
Conferences

48,758
Journals

25,554
Institutions

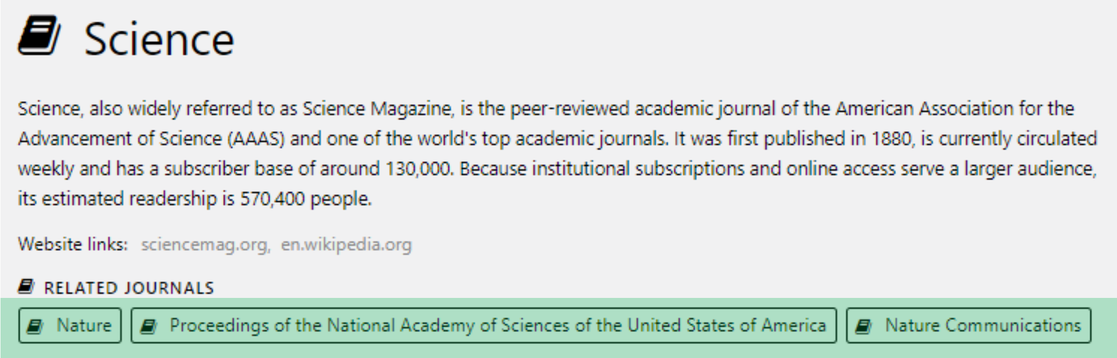
Microsoft Academic Graph

➪ metapath2vec ➪



- https://academic.microsoft.com/
- https://www.openacademic.ai/oag/
- metapath2vec: scalable representation learning for heterogeneous networks. In *KDD* 2017.

Application 1: Related Venues



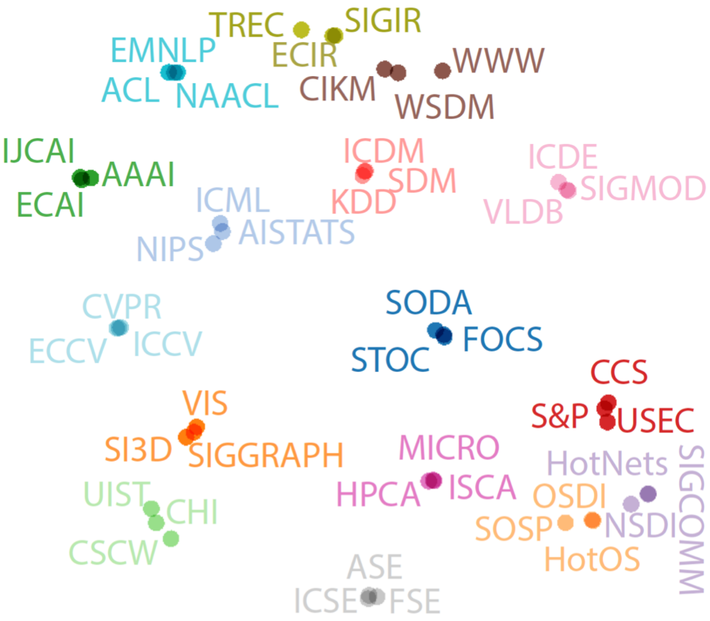
Science

Science, also widely referred to as Science Magazine, is the peer-reviewed academic journal of the American Association for the Advancement of Science (AAAS) and one of the world's top academic journals. It was first published in 1880, is currently circulated weekly and has a subscriber base of around 130,000. Because institutional subscriptions and online access serve a larger audience, its estimated readership is 570,400 people.

Website links: sciencemag.org, en.wikipedia.org

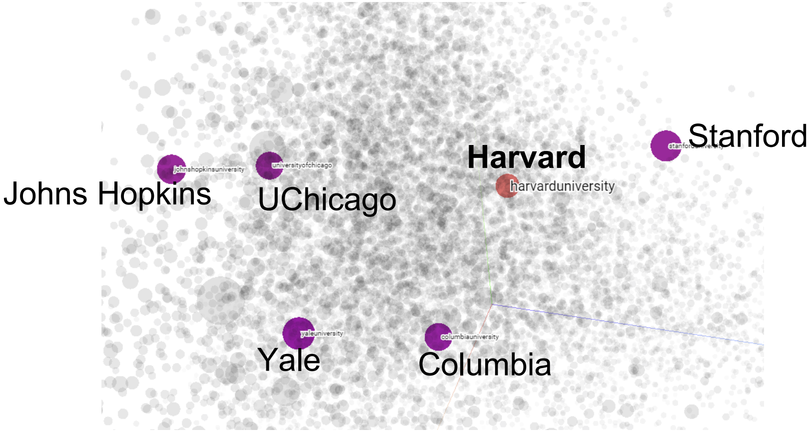
RELATED JOURNALS

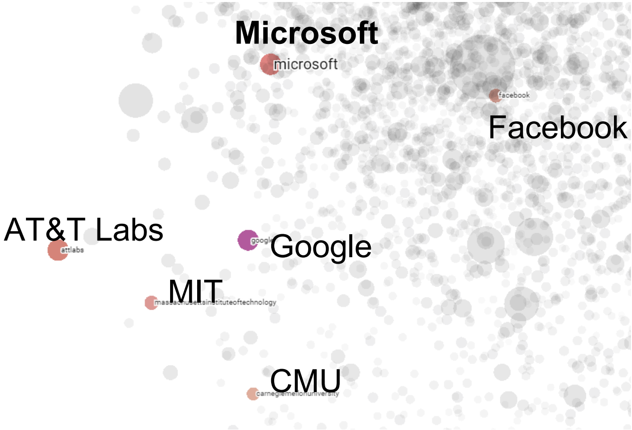
Nature | Proceedings of the National Academy of Sciences of the United States of America | Nature Communications



- https://academic.microsoft.com/
- https://www.openacademic.ai/oag/
- metapath2vec: scalable representation learning for heterogeneous networks. In *KDD* 2017.

Application 2: Similarity Search (Institution)





- https://academic.microsoft.com/
- https://www.openacademic.ai/oag/
- metapath2vec: scalable representation learning for heterogeneous networks. In *KDD* 2017.

What are the fundamentals

underlying random-walk + skip-gram based network embedding models?

Unifying DeepWalk, LINE, PTE, & node2vec as Matrix Factorization

- DeepWalk $$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$
- LINE $$\log\left(\frac{\mathrm{vol}(G)}{b}\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)$$
- PTE $$\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathsf{ww}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{ww}})^{-1}\boldsymbol{A}_{\mathsf{ww}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{ww}})^{-1}\\ \beta\,\mathrm{vol}(G_{\mathsf{dw}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{dw}})^{-1}\boldsymbol{A}_{\mathsf{dw}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{dw}})^{-1}\\ \gamma\,\mathrm{vol}(G_{\mathsf{lw}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{lw}})^{-1}\boldsymbol{A}_{\mathsf{lw}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{lw}})^{-1}\end{bmatrix}\right)-\log b$$
- node2vec $$\log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_{u}\boldsymbol{X}_{w,u}\underline{\boldsymbol{P}}_{c,w,u}^{r}+\sum_{u}\boldsymbol{X}_{c,u}\underline{\boldsymbol{P}}_{w,c,u}^{r}\right)}{b\left(\sum_{u}\boldsymbol{X}_{w,u}\right)\left(\sum_{u}\boldsymbol{X}_{c,u}\right)}\right)$$

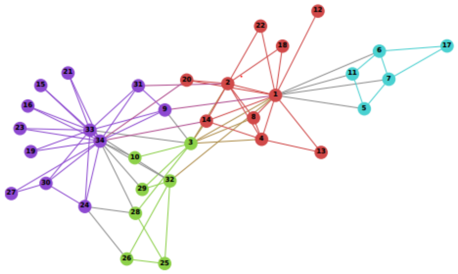
$\boldsymbol{A}$	Adjacency matrix
$\boldsymbol{D}$	Degree matrix

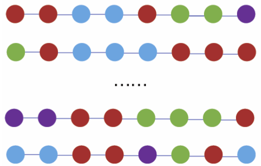
$$vol(G)=\sum_{i}\sum_{j}A_{ij}$$

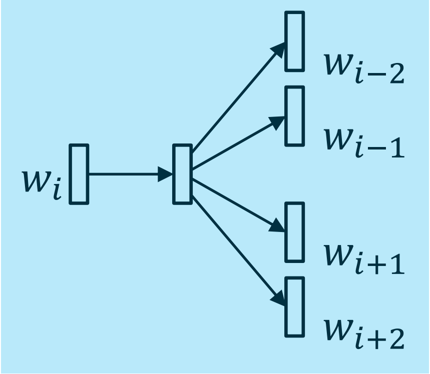
b: #negative samples
T: context window size

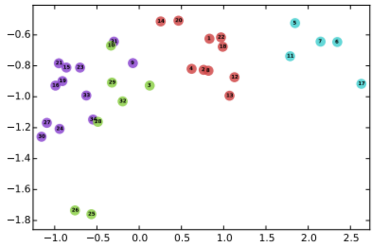
1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18*.

Understanding random walk + skip gram









$G = (V, E)$

- Adjacency matrix $\boldsymbol{A}$
- Degree matrix $\boldsymbol{D}$
- Volume of G: $vol(G)$

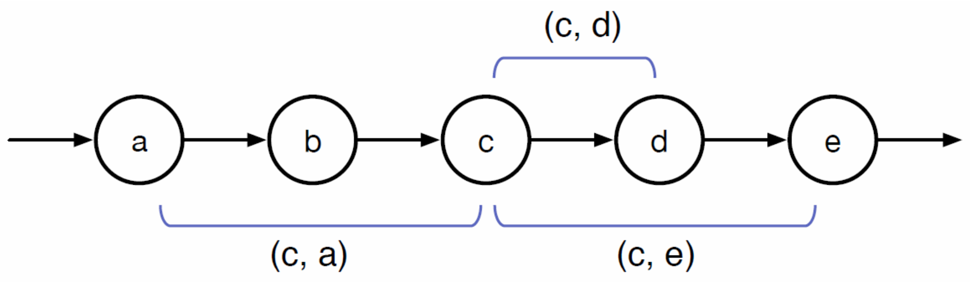
?

$$\log\left(\frac{\#(\boldsymbol{w}, \boldsymbol{c})|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

- #(w,c): co-occurrence of w & c
- #(w): occurrence of word w
- #(c): occurrence of context c
- $|\mathcal{D}|$: number of word-context pairs

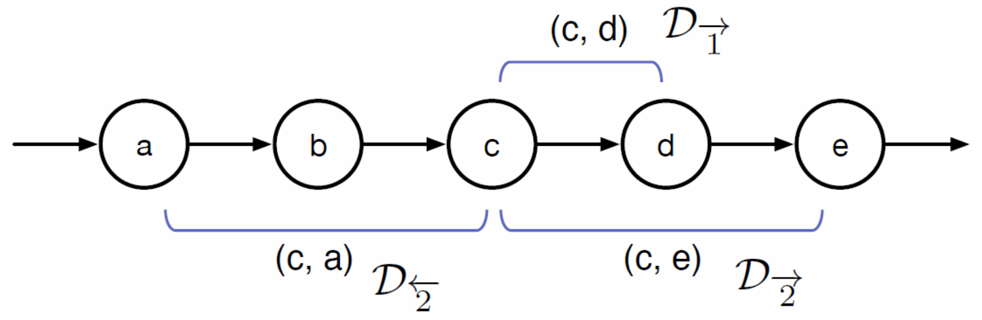
Levy and Goldberg. Neural word embeddings as implicit matrix factorization. In *NIPS 2014*

Understanding random walk + skip gram



Suppose the multiset $\mathcal{D}$ is constructed based on random walk on graphs, can we interpret $log\frac{\#(w,c)|\mathcal{D}|}{b\#(w)\#(c)}$ with graph structures?

Understanding random walk + skip gram



- Partition the multiset $\mathcal{D}$ into several sub-multisets according to the way in which each node and its context appear in a random walk node sequence.
- More formally, for $r = 1, 2, \cdots, T$, we define

$$\mathcal{D}_{\overrightarrow{r}} = \left\{(w, c) : (w, c) \in \mathcal{D}, w = w_j^n, c = w_{j+r}^n\right\}$$

$$\mathcal{D}_{\overleftarrow{r}} = \left\{(w, c) : (w, c) \in \mathcal{D}, w = w_{j+r}^n, c = w_j^n\right\}$$

Distinguish direction and distance

Understanding random walk + skip gram

$$\log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\cdot\#(c)}\right)=\log\left(\frac{\frac{\#(w,c)}{|\mathcal{D}|}}{b\frac{\#(w)}{|\mathcal{D}|}\frac{\#(c)}{|\mathcal{D}|}}\right)$$

the length of random walk $L \rightarrow \infty$

$$\frac{\#(w,c)}{|\mathcal{D}|}=\frac{1}{2T}\sum_{r=1}^{T}\left(\frac{\#(w,c)_{\vec{r}}}{|\mathcal{D}_{\vec{r}}|}+\frac{\#(w,c)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|}\right)$$

$$\frac{\#(w,c)_{\vec{r}}}{|\mathcal{D}_{\vec{r}}|}\xrightarrow{p}\frac{d_w}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{w,c}$$

$$\frac{\#(w,c)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|}\xrightarrow{p}\frac{d_c}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{c,w}$$

$$\frac{\#(w,c)}{|\mathcal{D}|}\xrightarrow{p}\frac{1}{2T}\sum_{r=1}^{T}\left(\frac{d_w}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{w,c}+\frac{d_c}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{c,w}\right)$$

$$\boldsymbol{P}=\boldsymbol{D}^{-1}\boldsymbol{A}$$

$$\frac{\#(w)}{|\mathcal{D}|}\xrightarrow{p}\frac{d_w}{\mathrm{vol}(G)}$$

$$\frac{\#(c)}{|\mathcal{D}|}\xrightarrow{p}\frac{d_c}{\mathrm{vol}(G)}$$

Understanding random walk + skip gram

$$\log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\cdot\#(c)}\right) = \log\left(\frac{\frac{\#(w,c)}{|\mathcal{D}|}}{b\frac{\#(w)}{|\mathcal{D}|}\frac{\#(c)}{|\mathcal{D}|}}\right)$$

the length of random walk $L \to \infty$

$$\frac{\#(w,c)}{|\mathcal{D}|} \xrightarrow{p} \frac{1}{2T}\sum_{r=1}^{T}\left(\frac{d_w}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{w,c} + \frac{d_c}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{c,w}\right)$$

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$$

$$\frac{\#(w)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_w}{\mathrm{vol}(G)} \qquad \frac{\#(c)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_c}{\mathrm{vol}(G)}$$

$$\frac{\#(w,c)\,|\mathcal{D}|}{\#(w)\cdot\#(c)} = \frac{\frac{\#(w,c)}{|\mathcal{D}|}}{\frac{\#(w)}{|\mathcal{D}|}\cdot\frac{\#(c)}{|\mathcal{D}|}} \xrightarrow{p} \frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\frac{d_w}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{w,c} + \frac{d_c}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{c,w}\right)}{\frac{d_w}{\mathrm{vol}(G)}\cdot\frac{d_c}{\mathrm{vol}(G)}}$$

$$= \frac{\mathrm{vol}(G)}{2T}\left(\frac{1}{d_c}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{w,c} + \frac{1}{d_w}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{c,w}\right)$$

Understanding random walk + skip gram

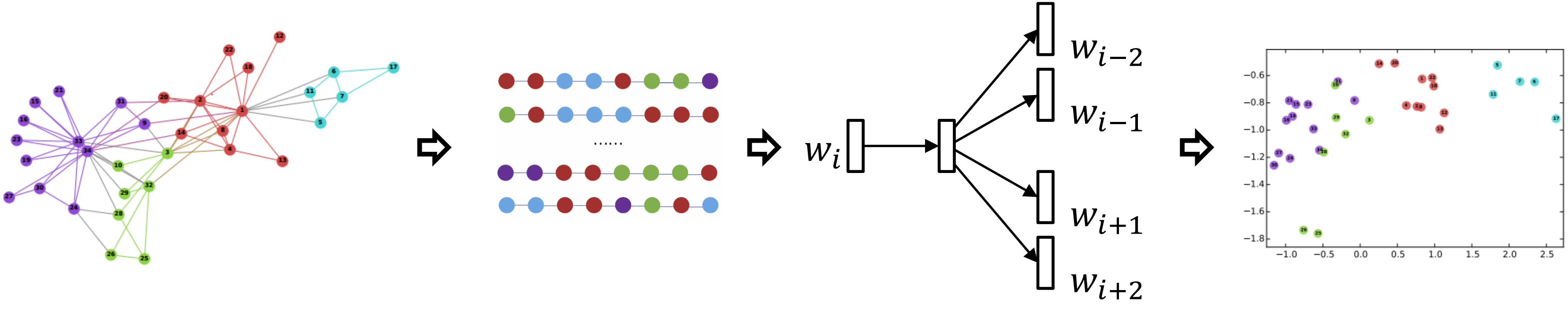
$$\frac{\#(w,c)\,|\mathcal{D}|}{\#(w)\cdot\#(c)} \xrightarrow{p} \frac{\mathrm{vol}(G)}{2T}\left(\frac{1}{d_c}\sum_{r=1}^{T}(\boldsymbol{P}^r)_{w,c} + \frac{1}{d_w}\sum_{r=1}^{T}(\boldsymbol{P}^r)_{c,w}\right)$$

$$\frac{\mathrm{vol}(G)}{2T}\left(\sum_{r=1}^{T}\boldsymbol{P}^r\boldsymbol{D}^{-1} + \sum_{r=1}^{T}\boldsymbol{D}^{-1}(\boldsymbol{P}^r)^\top\right)$$

$$= \frac{\mathrm{vol}(G)}{2T}\left(\sum_{r=1}^{T}\underbrace{\boldsymbol{D}^{-1}\boldsymbol{A}\times\cdots\times\boldsymbol{D}^{-1}\boldsymbol{A}}_{r\text{ terms}}\boldsymbol{D}^{-1} + \sum_{r=1}^{T}\boldsymbol{D}^{-1}\underbrace{\boldsymbol{A}\boldsymbol{D}^{-1}\times\cdots\times\boldsymbol{A}\boldsymbol{D}^{-1}}_{r\text{ terms}}\right)$$

$$= \frac{\mathrm{vol}(G)}{T}\sum_{r=1}^{T}\underbrace{\boldsymbol{D}^{-1}\boldsymbol{A}\times\cdots\times\boldsymbol{D}^{-1}\boldsymbol{A}}_{r\text{ terms}}\boldsymbol{D}^{-1} = \mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}\boldsymbol{P}^r\right)\boldsymbol{D}^{-1}.$$

Understanding random walk + skip gram



DeepWalk is asymptotically and implicitly factorizing

$$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

$vol(G) = \sum_i \sum_j A_{ij}$

b: #negative samples

T: context window size

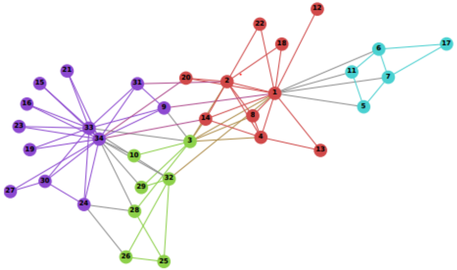
1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18*.

Unifying DeepWalk, LINE, PTE, & node2vec as Matrix Factorization

- DeepWalk $\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$
- LINE $\log\left(\frac{\mathrm{vol}(G)}{b}\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)$
- PTE $\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathsf{ww}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{ww}})^{-1}\boldsymbol{A}_{\mathsf{ww}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{ww}})^{-1}\\ \beta\,\mathrm{vol}(G_{\mathsf{dw}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{dw}})^{-1}\boldsymbol{A}_{\mathsf{dw}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{dw}})^{-1}\\ \gamma\,\mathrm{vol}(G_{\mathsf{lw}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{lw}})^{-1}\boldsymbol{A}_{\mathsf{lw}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{lw}})^{-1}\end{bmatrix}\right)-\log b$
- node2vec $\log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_{u}\boldsymbol{X}_{w,u}\underline{\boldsymbol{P}}_{c,w,u}^{r}+\sum_{u}\boldsymbol{X}_{c,u}\underline{\boldsymbol{P}}_{w,c,u}^{r}\right)}{b\left(\sum_{u}\boldsymbol{X}_{w,u}\right)\left(\sum_{u}\boldsymbol{X}_{c,u}\right)}\right)$

Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.* **The most cited paper in WSDM'18 as of May 2019**

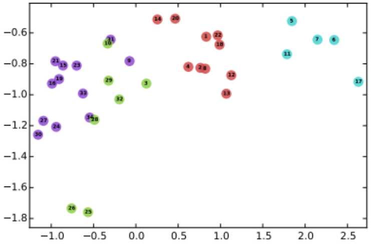
NetMF: explicitly factorizing the DeepWalk matrix



⇨

Matrix Factorization

⇨



DeepWalk is asymptotically and implicitly factorizing

$$\log\left(\frac{\operatorname{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*

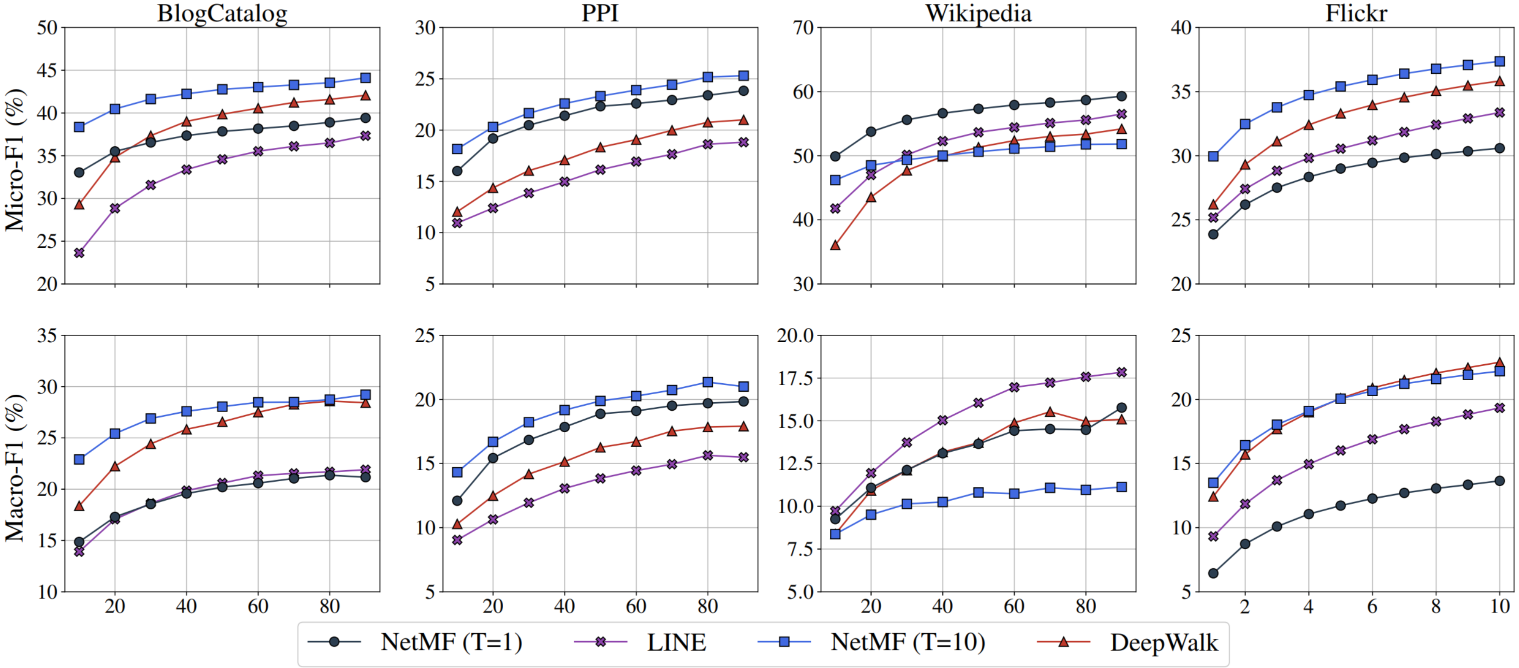
NetMF

1. Construction
2. Factorization

$$\boldsymbol{S} = \log\left(\frac{\operatorname{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*

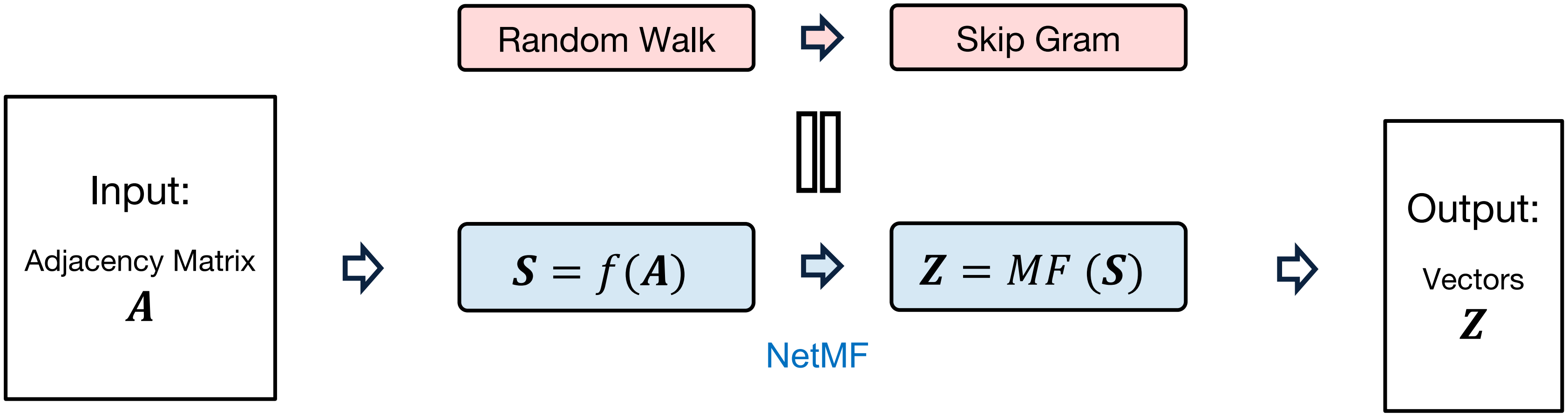
Results



Predictive performance on varying the ratio of training data;
The *x*-axis represents the ratio of labeled data (%)

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*

Connecting NE with graph neural networks



Random Walk ⇨ Skip Gram

$\Vert$

Input:

Adjacency Matrix

$\boldsymbol{A}$

⇨ $\boldsymbol{S} = f(\boldsymbol{A})$ ⇨ $\boldsymbol{Z} = MF(\boldsymbol{S})$ ⇨

Output:

Vectors

$\boldsymbol{Z}$

NetMF

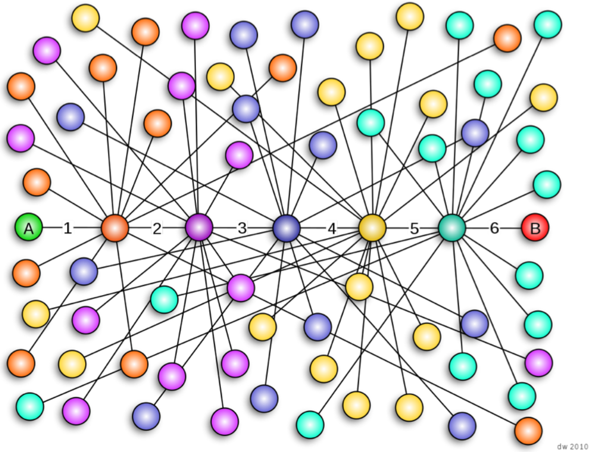
Incorporate network structures $\boldsymbol{A}$ into the similarity matrix $\boldsymbol{S}$, and then factorize $\boldsymbol{S}$

$$f(\boldsymbol{A}) = \log\left(\frac{\operatorname{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

---

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*

Challenges



⇨ $$\boldsymbol{S} = \log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

dense

NetMF is not practical for very large networks

NetMF

How can we solve this issue?

1. Construction
2. Factorization

$$\boldsymbol{S} = \log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

---

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**

NetSMF--Sparse

How can we solve this issue?

1. **Sparse** Construction
2. **Sparse** Factorization

$$\boldsymbol{S} = \log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**

Sparsify $\boldsymbol{S}$

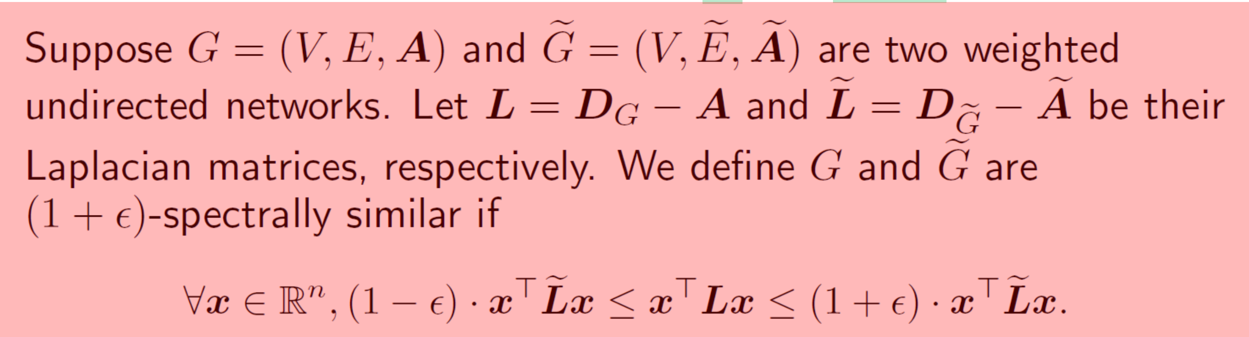
For random-walk matrix polynomial $\boldsymbol{L} = \boldsymbol{D} - \sum_{r=1}^{T} \alpha_r \boldsymbol{D} \left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^r$

where $\sum_{r=1}^{T} \alpha_r = 1$ and α_r non-negative

One can construct a $(1+\epsilon)$-spectral sparsifier $\tilde{\boldsymbol{L}}$ with $O(n \log n \epsilon^{-2})$ non-zeros

in time $O(T^2 m \epsilon^{-2} \log^2 n)$

$O(T^2 m \epsilon^{-2} \log n)$ for undirected graphs



Suppose $G = (V, E, \boldsymbol{A})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\boldsymbol{A}})$ are two weighted undirected networks. Let $\boldsymbol{L} = \boldsymbol{D}_G - \boldsymbol{A}$ and $\widetilde{\boldsymbol{L}} = \boldsymbol{D}_{\widetilde{G}} - \widetilde{\boldsymbol{A}}$ be their Laplacian matrices, respectively. We define G and $\widetilde{G}$ are $(1+\epsilon)$-spectrally similar if

$$\forall \boldsymbol{x} \in \mathbb{R}^n, (1-\epsilon) \cdot \boldsymbol{x}^\top \widetilde{\boldsymbol{L}} \boldsymbol{x} \leq \boldsymbol{x}^\top \boldsymbol{L} \boldsymbol{x} \leq (1+\epsilon) \cdot \boldsymbol{x}^\top \widetilde{\boldsymbol{L}} \boldsymbol{x}.$$

- Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng, Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification, COLT 2015.
- Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng. Spectral sparsification of random-walk matrix polynomials. arXiv:1502.03496.

Sparsify $\boldsymbol{S}$

For random-walk matrix polynomial $\boldsymbol{L} = \boldsymbol{D} - \sum_{r=1}^{T} \alpha_r \boldsymbol{D} \left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^r$

where $\sum_{r=1}^{T} \alpha_r = 1$ and α_r non-negative

One can construct a $(1+\epsilon)$-spectral sparsifier $\tilde{\boldsymbol{L}}$ with $O(n \log n \epsilon^{-2})$ non-zeros

in time $O(T^2 m \epsilon^{-2} \log^2 n)$

$$\alpha_1 = \cdots = \alpha_T = \frac{1}{T}$$

⇨

$$\begin{aligned}\boldsymbol{S} = &\log^{\circ}\left(\frac{\operatorname{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^r\right)\boldsymbol{D}^{-1}\right)\\ =&\log^{\circ}\left(\frac{\operatorname{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\boldsymbol{L})\boldsymbol{D}^{-1}\right)\\ \approx&\log^{\circ}\left(\frac{\operatorname{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\widetilde{\boldsymbol{L}})\boldsymbol{D}^{-1}\right)\end{aligned}$$

---

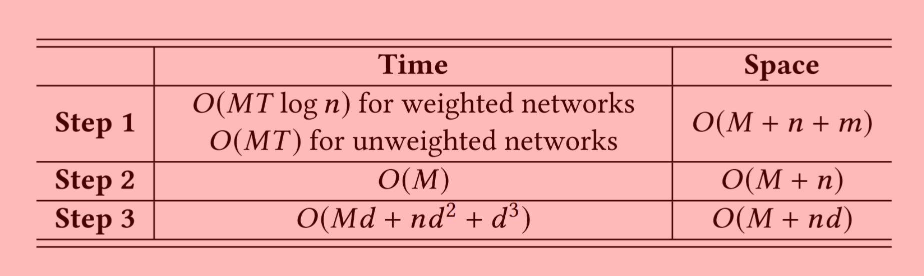
1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**

NetSMF --- Sparse

- Construct a random walk matrix polynomial sparsifier, $\widetilde{\boldsymbol{L}}$
- Construct a NetMF matrix sparsifier.

$$\operatorname{trunc\_log}^{\circ}\left(\frac{\operatorname{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\widetilde{\boldsymbol{L}})\boldsymbol{D}^{-1}\right)$$

- Factorize the constructed matrix



	Time	**Space**
Step 1	$O(MT \log n)$ for weighted networks $O(MT)$ for unweighted networks	$O(M+n+m)$
Step 2	$O(M)$	$O(M+n)$
Step 3	$O(Md+nd^2+d^3)$	$O(M+nd)$

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**

NetSMF---bounded approximation error

$$
\begin{aligned}
&\log^\circ\left(\frac{\operatorname{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^r\right)\boldsymbol{D}^{-1}\right)\\
=&\log^\circ\left(\frac{\operatorname{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\boldsymbol{L})\boldsymbol{D}^{-1}\right) \longrightarrow \boldsymbol{M}\\
\approx&\log^\circ\left(\frac{\operatorname{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\widetilde{\boldsymbol{L}})\boldsymbol{D}^{-1}\right) \longrightarrow \widetilde{\boldsymbol{M}}
\end{aligned}
$$

Theorem

The singular value of $\widetilde{\boldsymbol{M}}-\boldsymbol{M}$ satisfies

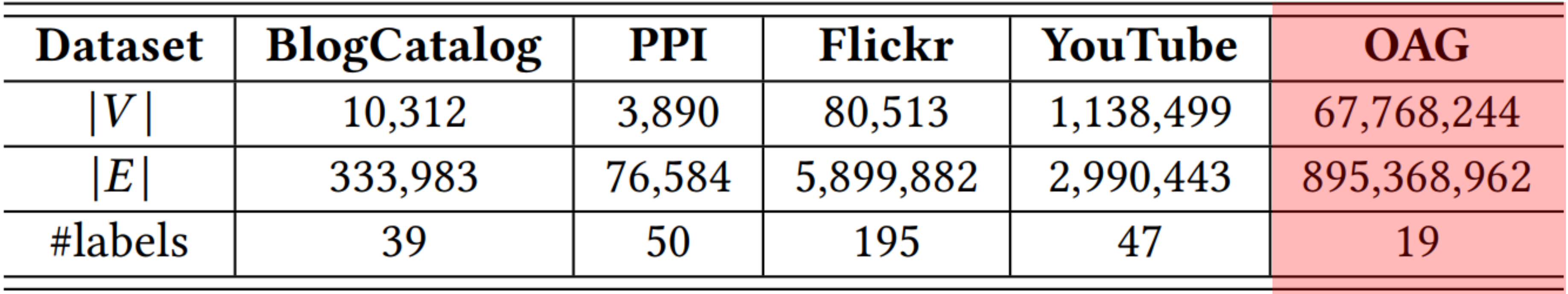
$$\sigma_i(\widetilde{\boldsymbol{M}}-\boldsymbol{M}) \leq \frac{4\epsilon}{\sqrt{d_i d_{\min}}}, \forall i \in [n].$$

Theorem

Let $\|\cdot\|_F$ be the matrix Frobenius norm. Then

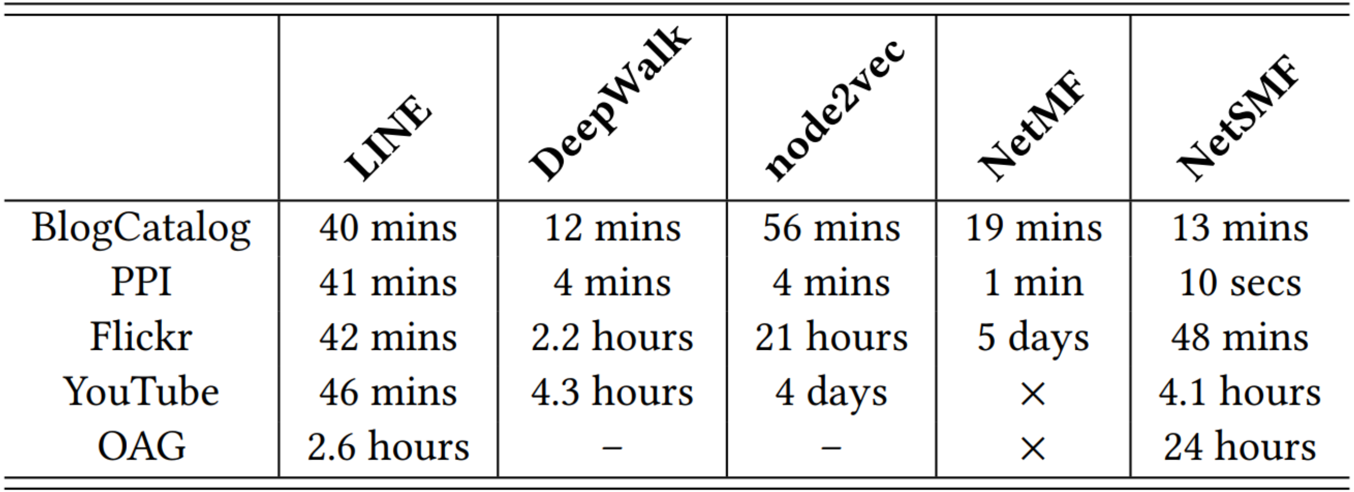
$$\left\|\operatorname{trunc\_log}^\circ\left(\frac{\operatorname{vol}(G)}{b}\widetilde{\boldsymbol{M}}\right)-\operatorname{trunc\_log}^\circ\left(\frac{\operatorname{vol}(G)}{b}\boldsymbol{M}\right)\right\|_F \leq \frac{4\epsilon\operatorname{vol}(G)}{b\sqrt{d_{\min}}}\sqrt{\sum_{i=1}^{n}\frac{1}{d_i}}.$$

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**



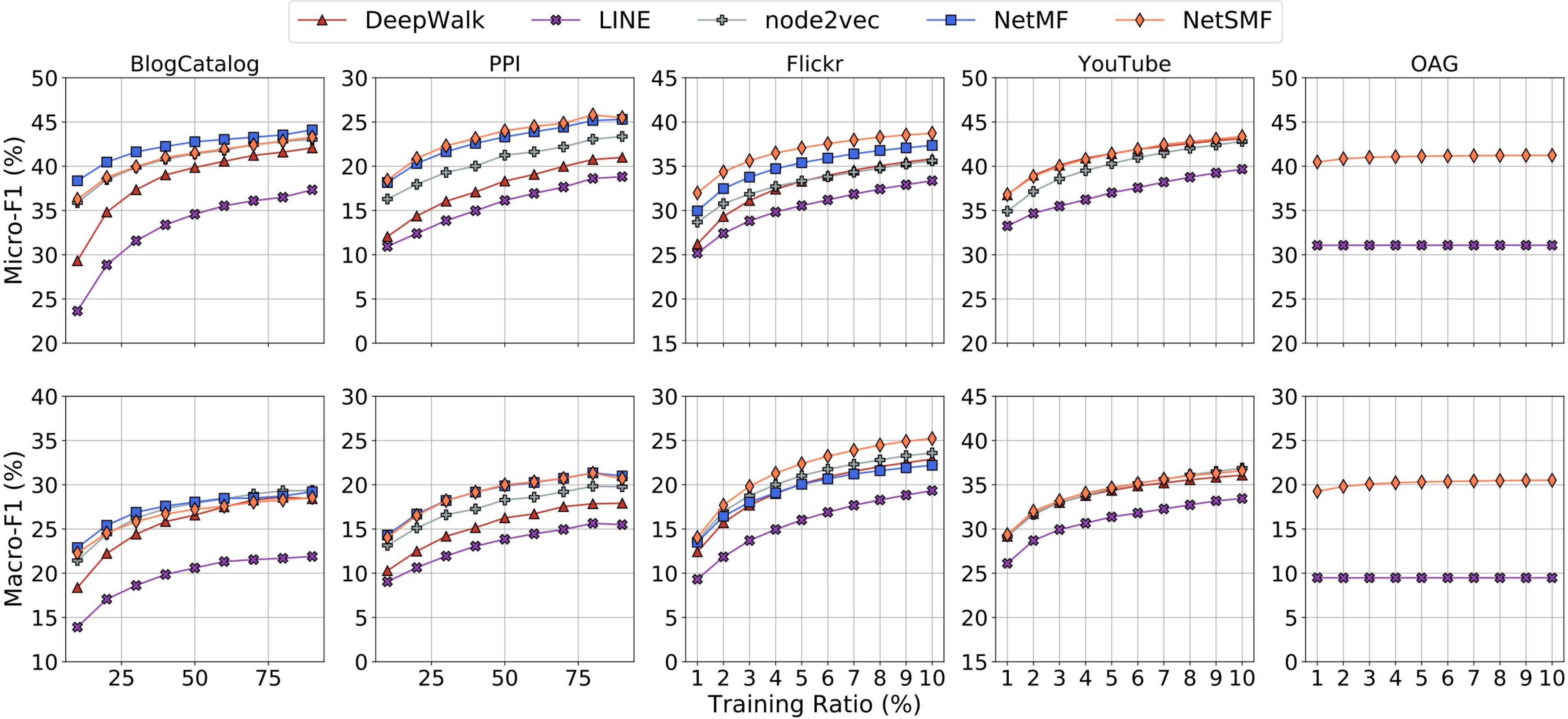
Dataset	BlogCatalog	PPI	Flickr	YouTube	OAG
$\|V\|$	10,312	3,890	80,513	1,138,499	67,768,244
$\|E\|$	333,983	76,584	5,899,882	2,990,443	895,368,962
#labels	39	50	195	47	19

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**

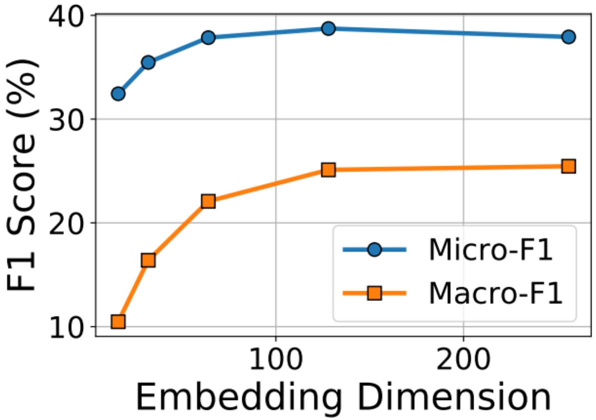


	LINE	**DeepWalk**	**node2vec**	**NetMF**	**NetSMF**
BlogCatalog	40 mins	12 mins	56 mins	19 mins	13 mins
PPI	41 mins	4 mins	4 mins	1 min	10 secs
Flickr	42 mins	2.2 hours	21 hours	5 days	48 mins
YouTube	46 mins	4.3 hours	4 days	×	4.1 hours
OAG	2.6 hours	–	–	×	24 hours

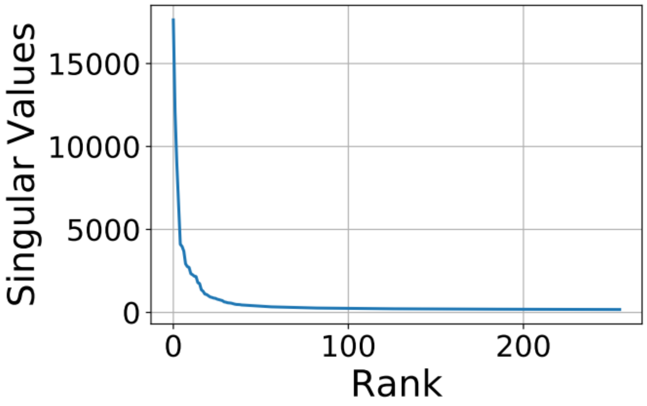
1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**



1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**



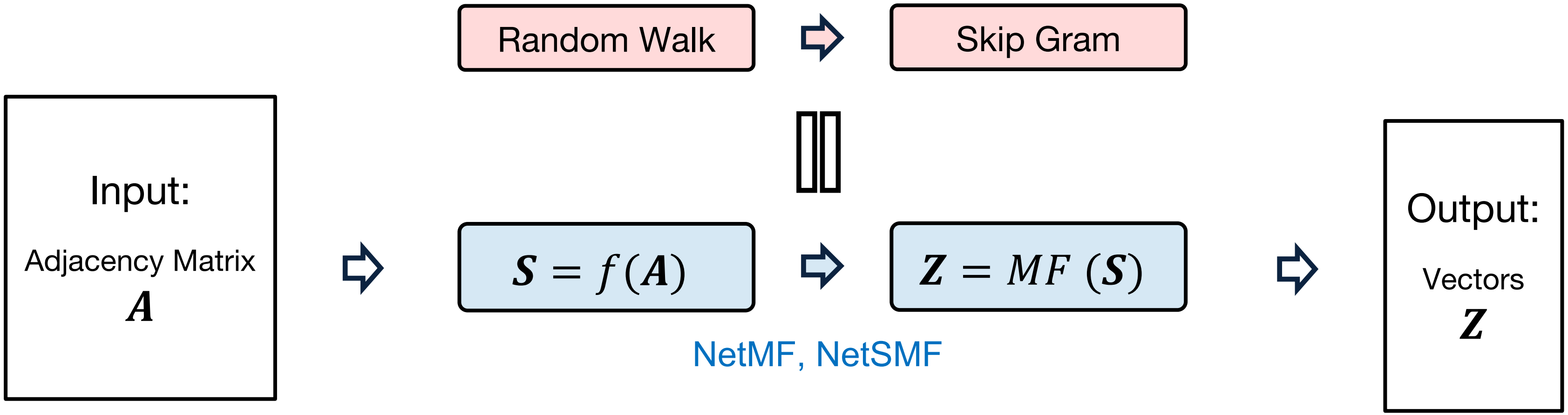
(a)



(b)

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019**

Connecting NE with graph neural networks



Input:

Adjacency Matrix

$\boldsymbol{A}$

Random Walk

Skip Gram

$\boldsymbol{S} = f(\boldsymbol{A})$

$\boldsymbol{Z} = MF(\boldsymbol{S})$

NetMF, NetSMF

Output:

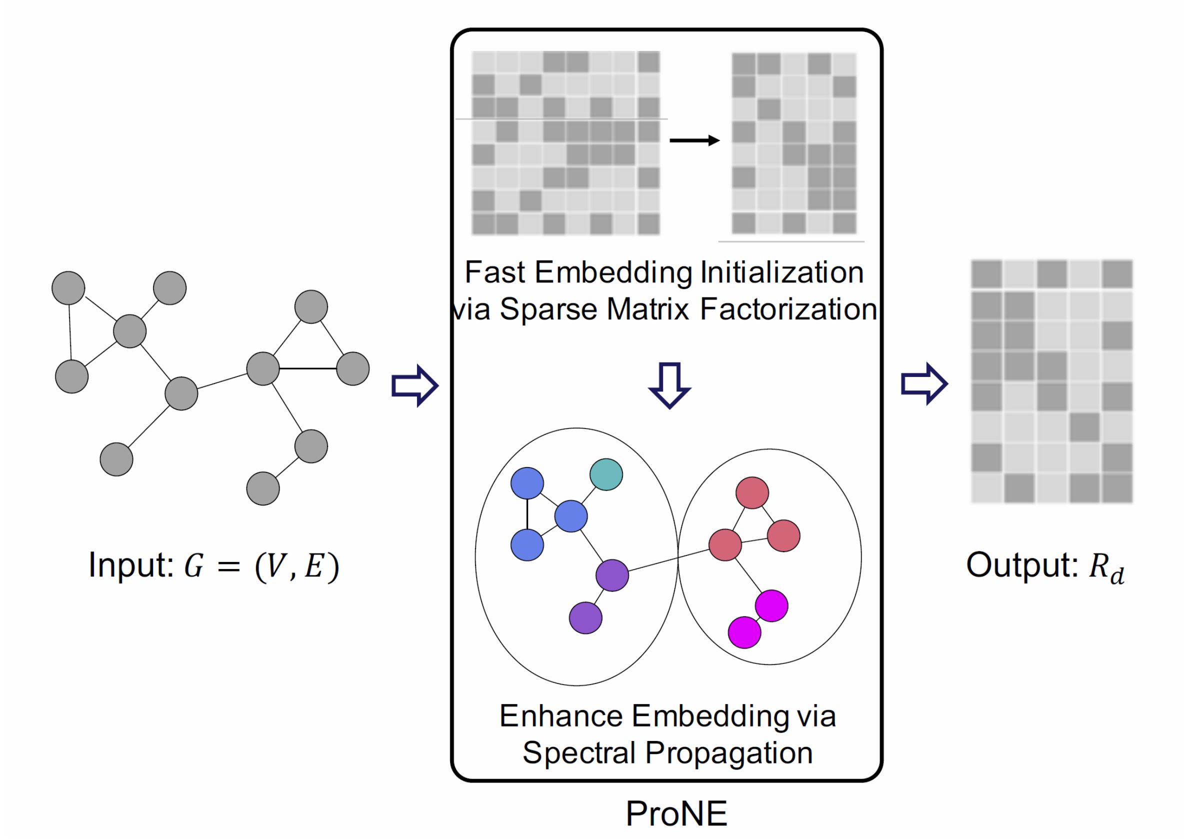
Vectors

$\boldsymbol{Z}$

Incorporate network structures $\boldsymbol{A}$ into the similarity matrix $\boldsymbol{S}$, and then factorize $\boldsymbol{S}$

$$f(\boldsymbol{A}) = \log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

ProNE: More fast & scalable network embedding



Input: $G = (V, E)$

Fast Embedding Initialization via Sparse Matrix Factorization

Enhance Embedding via Spectral Propagation

ProNE

Output: R_d

---

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Embedding enhancement via spectral propagation

$$R_d \leftarrow D^{-1}A(I_n - \tilde{L})\, R_d$$

$\widetilde{L} = Ug(\Lambda)U^T$ is the spectral filter of $L = I_n - D^{-1}A$

$D^{-1}A(I_n - \tilde{L})$ is $D^{-1}A$ modulated by the filter in the spectrum

---

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Chebyshev expansion for efficiency

- To avoid explicit eigendecomposition and Fourier transform
    - Chebyshev expansion $T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x)$ with $T_0(x) = 1, T_1(x) = x$

$$\begin{aligned} \widetilde{L} &= U diag([g(\lambda_1), ..., g(\lambda_n)])U^T \\ &\approx U \sum_{i=0}^{k-1} c_i(\theta) T_i(\bar{\Lambda}) U^T \\ &= \sum_{i=0}^{k-1} c_i(\theta) T_i(\bar{L}) \end{aligned}$$

⇨

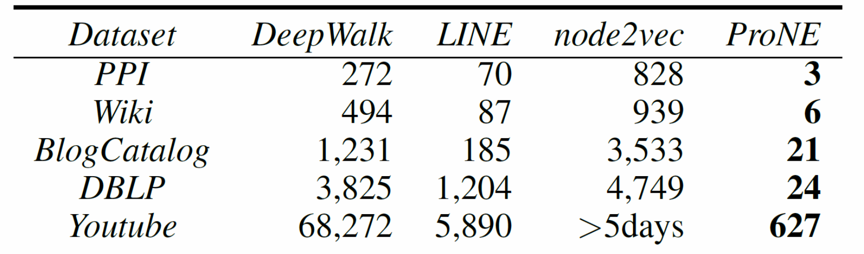
$$\widetilde{L} \approx B_0(\theta)T_0(\bar{L}) + 2\sum_{i=1}^{k-1}(-)^i B_i(\theta)T_i(\bar{L})$$

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Efficiency

20 Threads

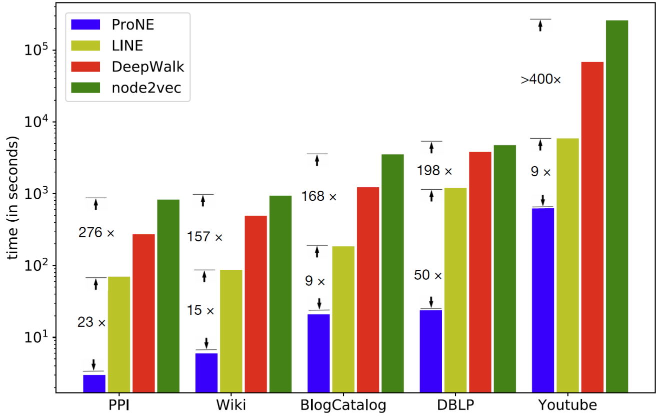
1 Thread



Dataset	*DeepWalk*	*LINE*	*node2vec*	*ProNE*
PPI	272	70	828	**3**
Wiki	494	87	939	**6**
BlogCatalog	1,231	185	3,533	**21**
DBLP	3,825	1,204	4,749	**24**
Youtube	68,272	5,890	>5days	**627**

19hours 98mins 10mins

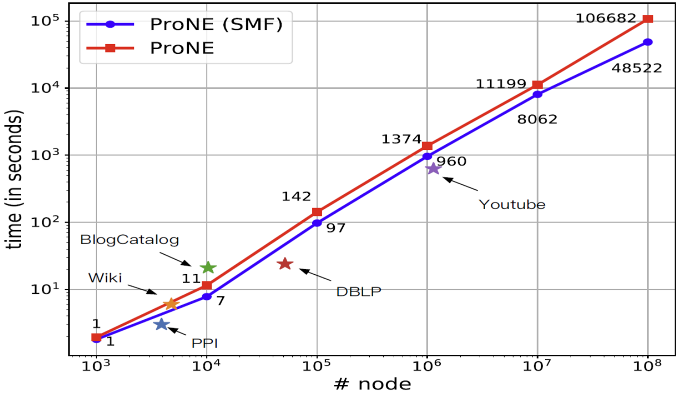
1.1M nodes



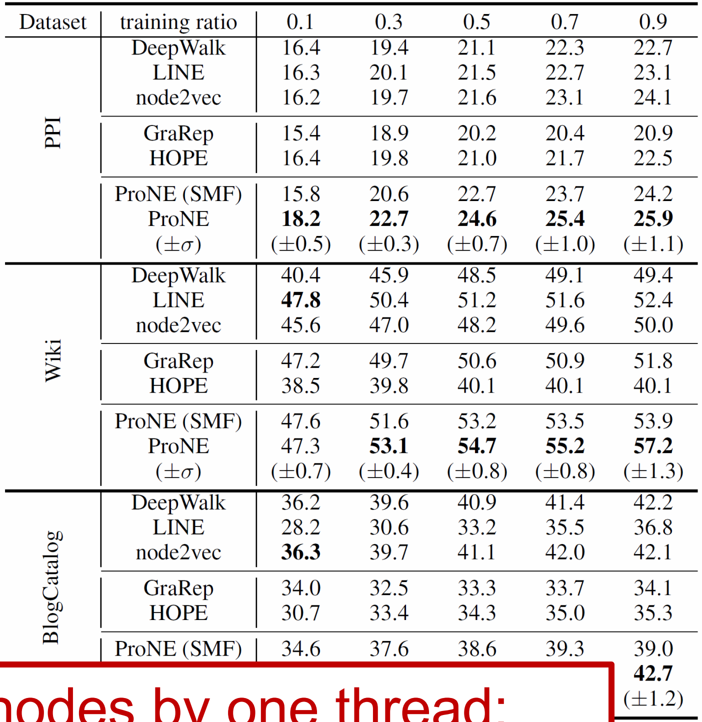
ProNE offers 10-400X speedups (1 thread vs 20 threads)

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Scalability & Effectiveness



(a) The node degree is fixed to 10 and #nodes grows

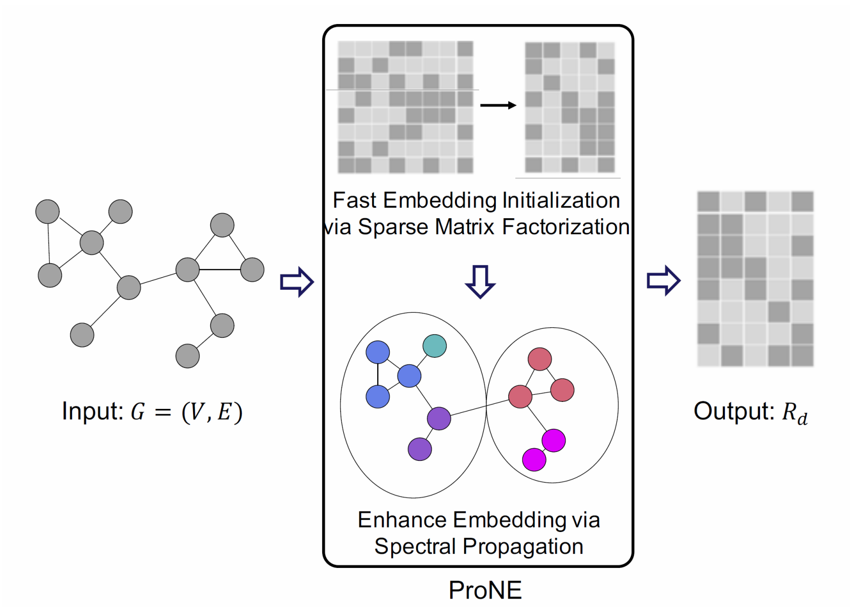


Dataset	training ratio	0.1	0.3	0.5	0.7	0.9
PPI	DeepWalk	16.4	19.4	21.1	22.3	22.7
	LINE	16.3	20.1	21.5	22.7	23.1
	node2vec	16.2	19.7	21.6	23.1	24.1
	GraRep	15.4	18.9	20.2	20.4	20.9
	HOPE	16.4	19.8	21.0	21.7	22.5
	ProNE (SMF)	15.8	20.6	22.7	23.7	24.2
	ProNE	**18.2**	**22.7**	**24.6**	**25.4**	**25.9**
	($\pm\sigma$)	($\pm$0.5)	($\pm$0.3)	($\pm$0.7)	($\pm$1.0)	($\pm$1.1)
Wiki	DeepWalk	40.4	45.9	48.5	49.1	49.4
	LINE	**47.8**	50.4	51.2	51.6	52.4
	node2vec	45.6	47.0	48.2	49.6	50.0
	GraRep	47.2	49.7	50.6	50.9	51.8
	HOPE	38.5	39.8	40.1	40.1	40.1
	ProNE (SMF)	47.6	51.6	53.2	53.5	53.9
	ProNE	47.3	**53.1**	**54.7**	**55.2**	**57.2**
	($\pm\sigma$)	($\pm$0.7)	($\pm$0.4)	($\pm$0.8)	($\pm$0.8)	($\pm$1.3)
BlogCatalog	DeepWalk	36.2	39.6	40.9	41.4	42.2
	LINE	28.2	30.6	33.2	35.5	36.8
	node2vec	**36.3**	39.7	41.1	42.0	42.1
	GraRep	34.0	32.5	33.3	33.7	34.1
	HOPE	30.7	33.4	34.3	35.0	35.3
	ProNE (SMF)	34.6	37.6	38.6	39.3	39.0
						42.7
						($\pm$1.2)

Embed 100,000,000 nodes by one thread:
29 hours with **performance superiority**

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Embedding enhancement



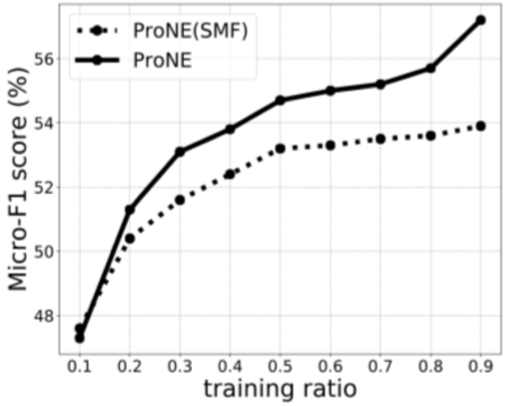
Fast Embedding Initialization via Sparse Matrix Factorization

Enhance Embedding via Spectral Propagation

Input: $G = (V, E)$

Output: R_d

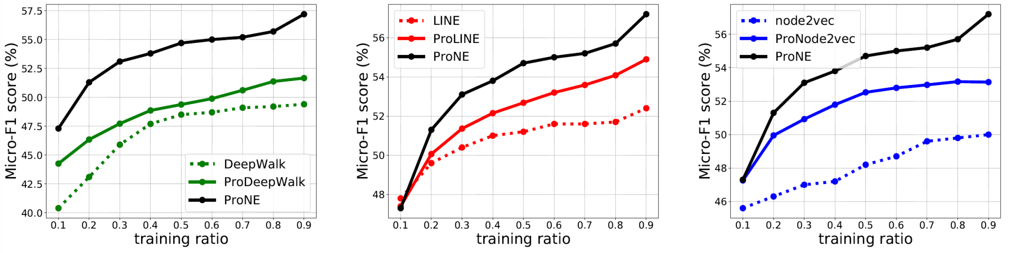
ProNE



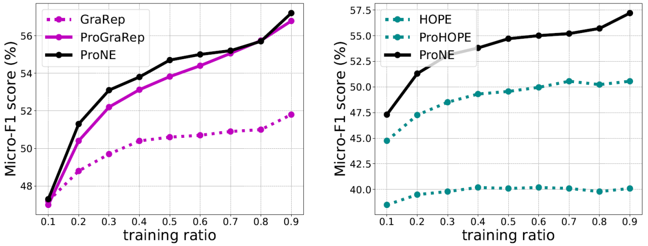
---

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

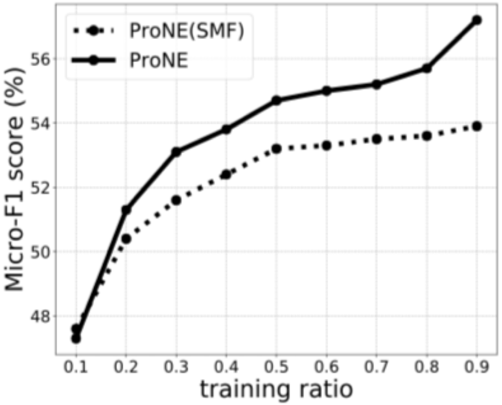
A general embedding enhancement framework



(a) ProDeepWalk (b) ProLINE (c) ProNode2vec

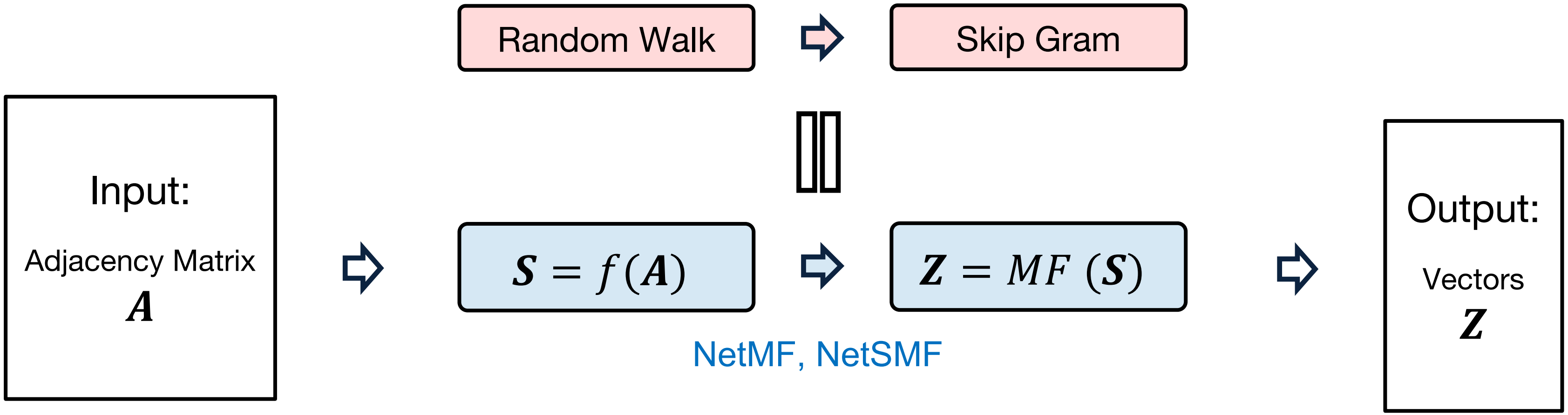


(d) ProGraRep (e) ProHOPE



1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Connecting NE with graph neural networks



Random Walk ⇨ Skip Gram

Input:

Adjacency Matrix

$\boldsymbol{A}$

⇨ $\boldsymbol{S} = f(\boldsymbol{A})$ ⇨ $\boldsymbol{Z} = MF(\boldsymbol{S})$ ⇨

Output:

Vectors

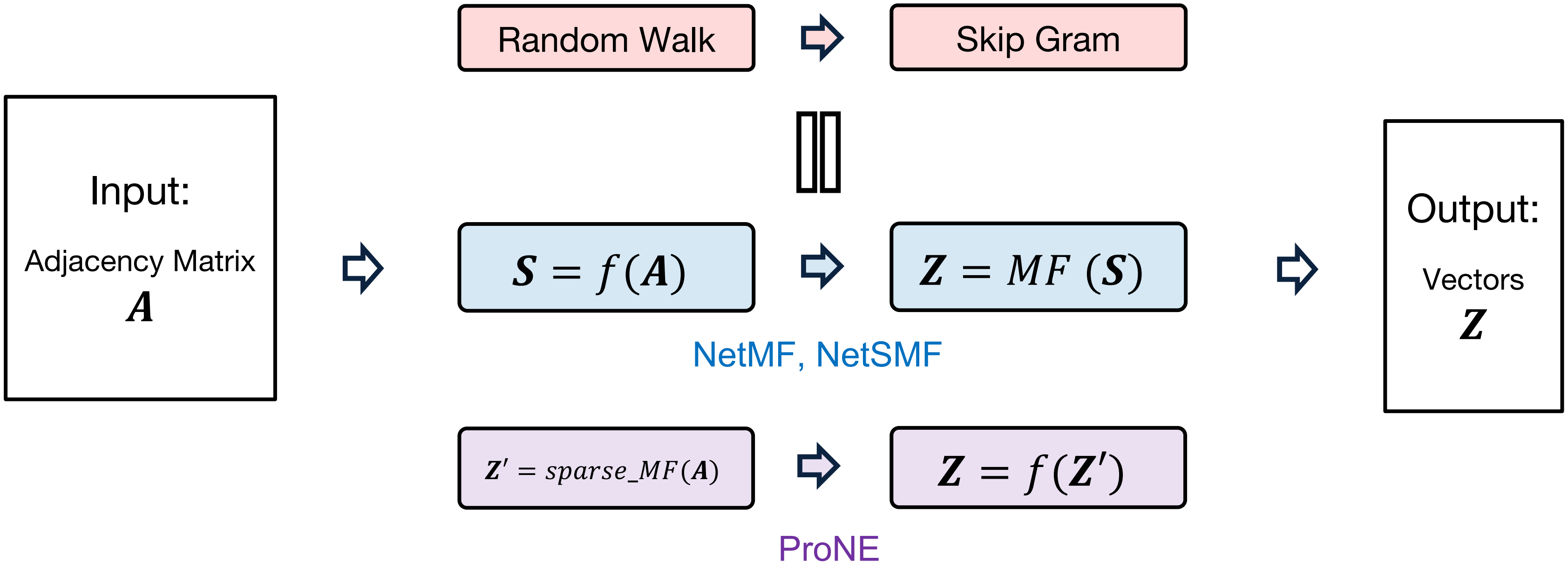
$\boldsymbol{Z}$

NetMF, NetSMF

Incorporate network structures $\boldsymbol{A}$ into the similarity matrix $\boldsymbol{S}$, and then factorize

$$f(\boldsymbol{A}) = \log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

Connecting NE with graph neural networks



Input:

Adjacency Matrix

$\boldsymbol{A}$

Random Walk ⇨ Skip Gram

$\parallel$

$\boldsymbol{S} = f(\boldsymbol{A})$ ⇨ $\boldsymbol{Z} = MF\ (\boldsymbol{S})$

NetMF, NetSMF

$\boldsymbol{Z}' = sparse\_MF(\boldsymbol{A})$ ⇨ $\boldsymbol{Z} = f(\boldsymbol{Z}')$

ProNE

Output:

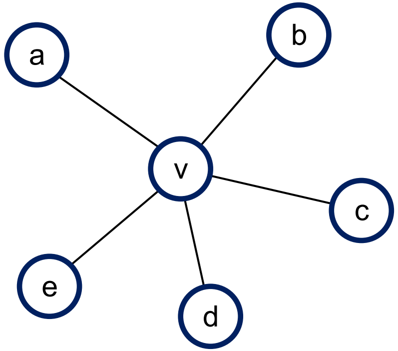
Vectors

$\boldsymbol{Z}$

Factorize $\boldsymbol{A}$, and then incorporate network structures via spectral propagation

Connecting NE with graph neural networks

ProNE: $R_d \leftarrow D^{-1}A(I_n - \tilde{L})\, R_d$



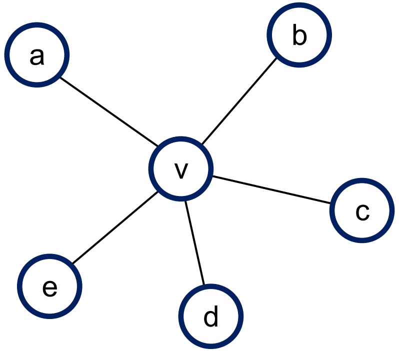
$$\boldsymbol{h}_v = f(\boldsymbol{h}_v, \boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

1. Defferrard et al. Convolutional Neural Networks on Graphs with Fast Locailzied Spectral Filtering. In **NIPS 2016**
2. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Graph Neural Networks

- Input: an undirected weighted network $G = (V, E)$ with $|V| = n$ & $|E| = \mathrm{m}$
  - Adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{n\times \mathrm{n}}$
  - $A_{i,j} = \begin{cases} a_{i,j} > 0 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$
  - Degree matrix $\boldsymbol{D} = diag(d_1, d_2, \cdots, d_n)$
  - Node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times q}$
- Output: for each node, its k-dimension latent feature representation vector $\mathbf{Z}^{n\times k}$
  - –Latent feature embedding matrix $\mathbf{Z} \in \mathbb{R}^{n\times k}$

The Core of Graph Neural Networks

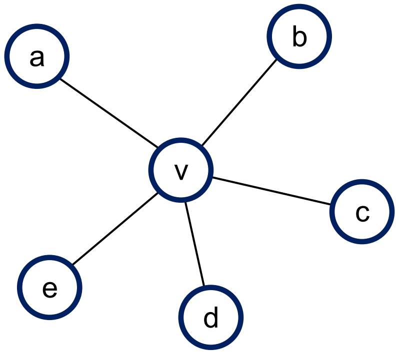


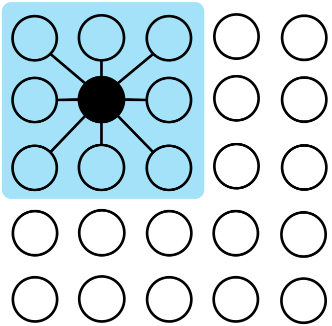
$$\boldsymbol{h}_v = f(\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

Neighborhood Aggregation:

Aggregate neighbor information and pass into a neural network

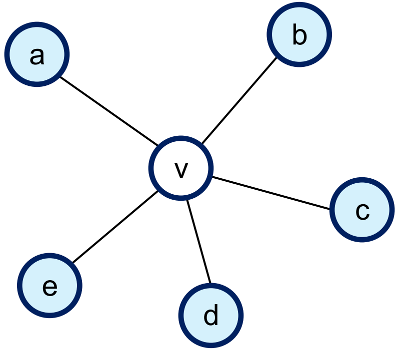
Graph neural networks





- **Neighborhood Aggregation:**
    - Aggregate neighbor information and pass into a neural network
    - It can be viewed as a center-surround filter in CNN---graph convolutions!

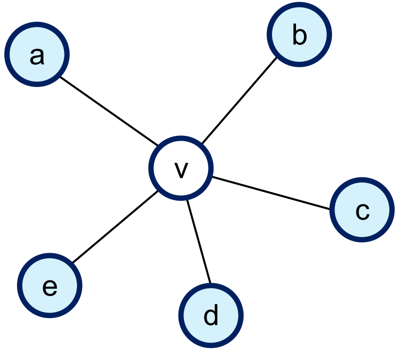
GNN: Graph convolutional networks



$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017
2. Defferrard et al. Convolutional Neural Networks on Graphs with Fast Locailzied Spectral Filtering. In **NIPS 2016**

GNN: Graph convolutional networks



parameters in layer k

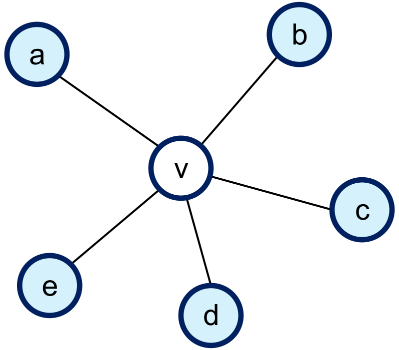
Non-linear activation function (e.g., ReLU)

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

node v's embedding at layer k

the neighbors of node v

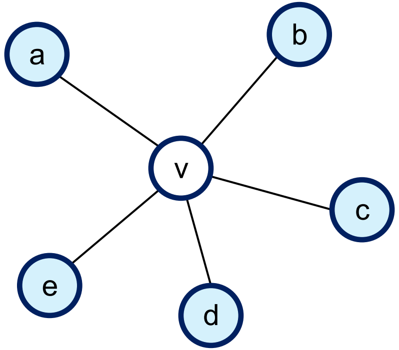
GNN: Graph convolutional networks



$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph convolutional networks



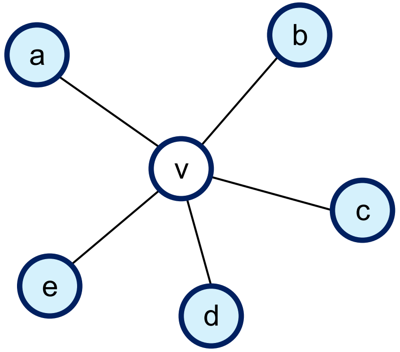
Aggregate from v's neighbors

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}})$$

Aggregate from itself

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph convolutional networks

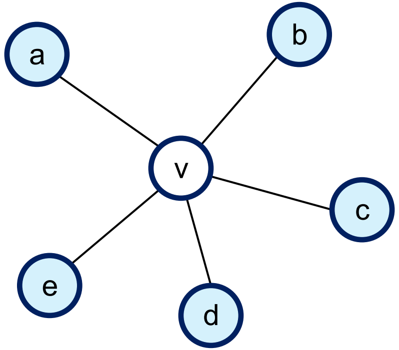


The same parameters for both its neighbors & itself

$$\boldsymbol{h}_v^k = \sigma\Big(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}}\Big)$$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph convolutional networks



$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

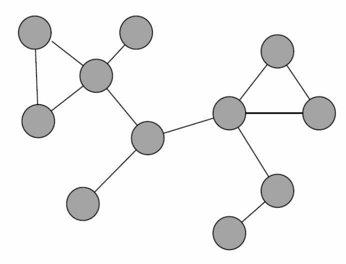
$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} +$$

$$\boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}})$$

$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{I}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph convolutional networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

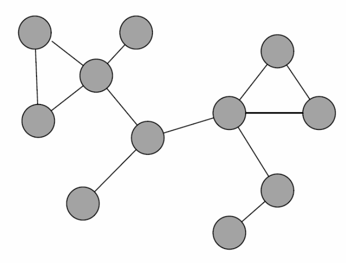
$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \quad \boldsymbol{Z} = \boldsymbol{H}^K$$

Input

Output

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph convolutional networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \quad \boldsymbol{Z} = \boldsymbol{H}^K$$

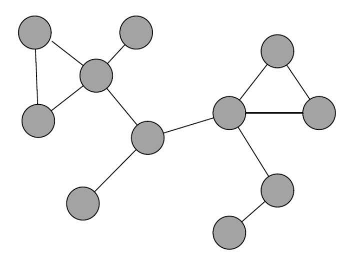
Input

Output

- Model training
  - The common setting is to have an end to end training framework with a supervised task
  - That is, define a loss function over $\boldsymbol{Z}$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph convolutional networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

⇨ $\boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$ ⇨ $\boldsymbol{Z} = \boldsymbol{H}^K$

Input

- Benefits: Parameter sharing for all nodes
  - #parameters is subline in |V|
  - Enable inductive learning for new nodes

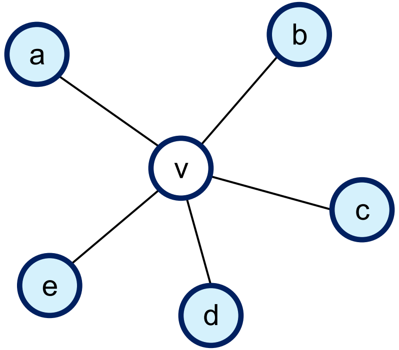
Output

GNN: Graph convolutional networks

$$\boldsymbol{H}^{k} = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$$

- GCN is one way of neighbor aggregations
- **GraphSage**
- Graph Attention
- … …

GraphSage



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

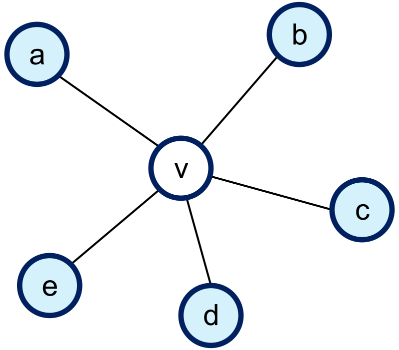
GraphSage

Instead of summation, it concatenates neighbor & self embeddings

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}^k \cdot \mathrm{AGG}\left(\{\boldsymbol{h}_u^{k-1}, \forall u \in N(v)\}\right), \boldsymbol{B}^k \boldsymbol{h}_v^{k-1}])$$

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017

GraphSage



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

GraphSage

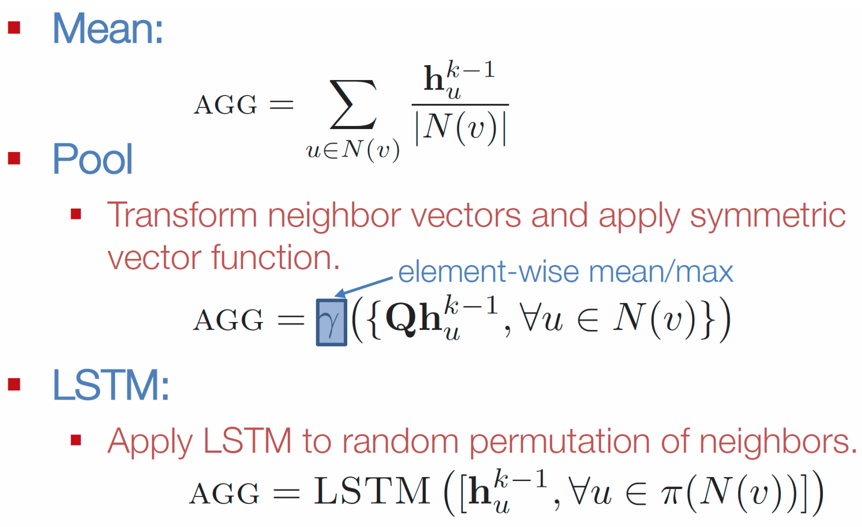
$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}^k \cdot \mathrm{AGG}\left(\{\boldsymbol{h}_u^{k-1}, \forall u \in N(v)\}\right), \boldsymbol{B}^k \boldsymbol{h}_v^{k-1}])$$

Generalized aggregation: any differentiable function that maps set of vectors to a single vector

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017

GraphSage

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}^k \cdot \mathrm{AGG}(\{\boldsymbol{h}_u^{k-1}, \forall u \in N(v)\}), \boldsymbol{B}^k \boldsymbol{h}_v^{k-1}])$$



- Mean:

$$\mathrm{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- Pool
  - Transform neighbor vectors and apply symmetric vector function.

element-wise mean/max

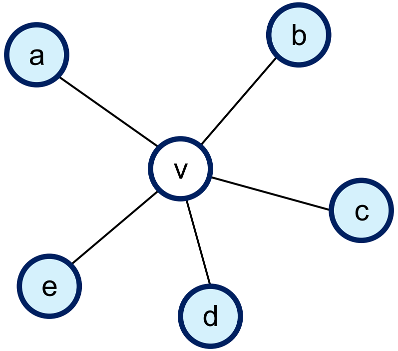
$$\mathrm{AGG} = \boxed{\gamma}\left(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

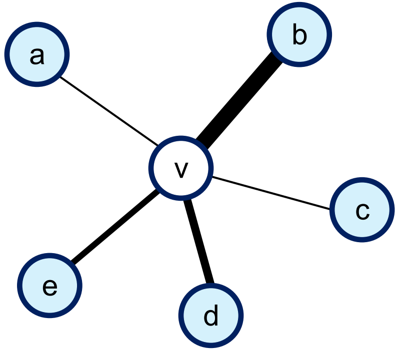
- LSTM:
  - Apply LSTM to random permutation of neighbors.

$$\mathrm{AGG} = \mathrm{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
Slide snipping from “Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning”

Graph Neural Networks





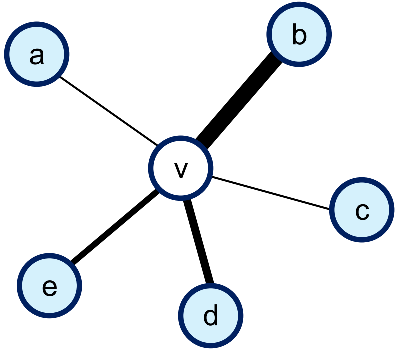
Velickovic et al. Graph Attention Networks. ICLR 2018

Graph Neural Networks

$$\boldsymbol{H}^{k} = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$$

- GCN is one way of neighbor aggregations
- GraphSage
- **Graph Attention**
- … …

GNN: Graph Attention



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

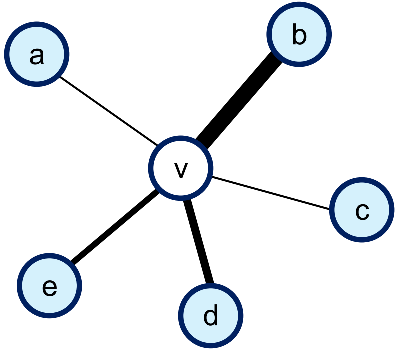
Graph Attention

$$\boldsymbol{h}_v^k = \sigma(\sum_{u \in \mathrm{N}(v) \cup v} \alpha_{v,u} \boldsymbol{W}^k \boldsymbol{h}_u^{k-1})$$

Learned attention weights

Velickovic et al. Graph Attention Networks. ICLR 2018

GNN: Graph Attention



$$\alpha_{v,u} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_u]\right)\right)}{\sum_{u' \in N(v) \cup \{v\}} \exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_{u'}]\right)\right)}$$

Various ways to define attention!

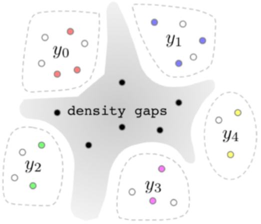
Velickovic et al. Graph Attention Networks. ICLR 2018

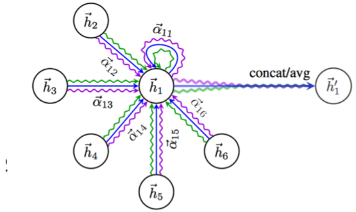
Graph neural networks

Graph Isomorphism Network, Deep Graph Infomax — 2019: Velickovic et al. & Xu et al., ICLR’19

Graph attention — 2018: Velickovic et al., ICLR’18

Neural message passing, GraphSage — 2017: Gilmer et al., ICML’17; Hamilton et al., NIPS’17





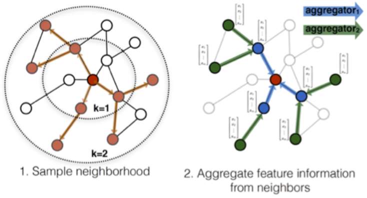
Gated graph neural network — 2016: Li et al., ICLR’16

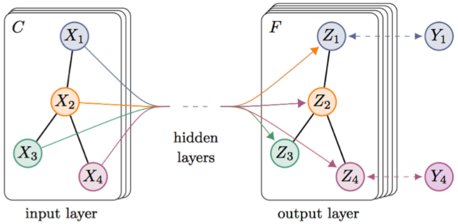
structure2vec — 2016: Dai et al., ICML’16

Graph convolutional network — 2015: Duvenaud et al., NIPS’15; Kipf & Welling ICLR’17

Spectral graph convolution — 2014: Bruna et al., ICLR’14

Graph neural network — 2005: Gori et al., IJCNN’05



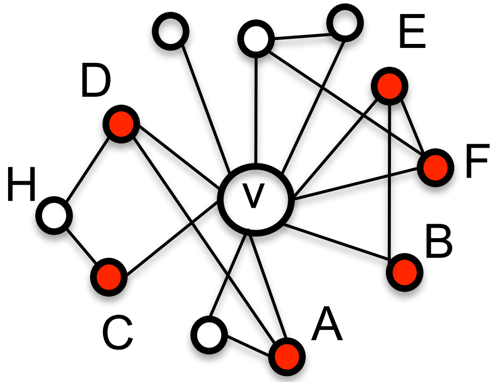


GNN applications & systems

- DeepInf: Modeling social influence with graph neural networks
- LinKG: Knowledge graph linking with heterogeneous graph attention
- AliGraph: A comprehensive graph neural network platform.
    - **Dr. Hongxia Yang**
    - Applied data science invited talk
    - 10AM--12PM, Thursday, Aug 6th
    - Cook Room, Street Level, Egan Center

1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.
2. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

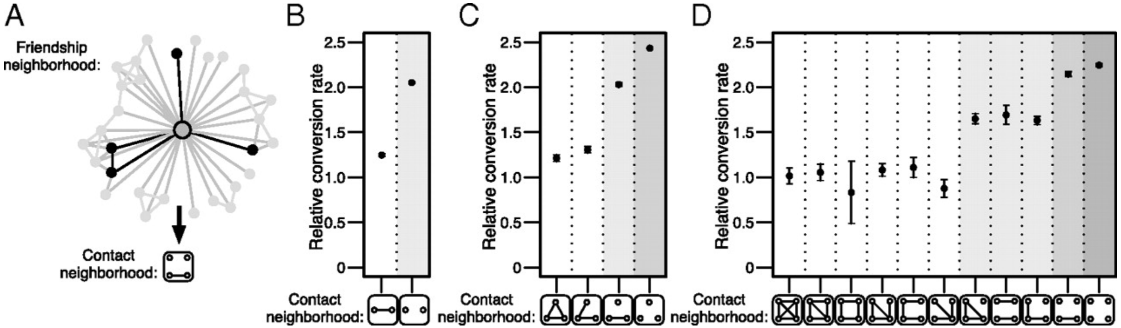
DeepInf: Modeling social influence with graph neural networks

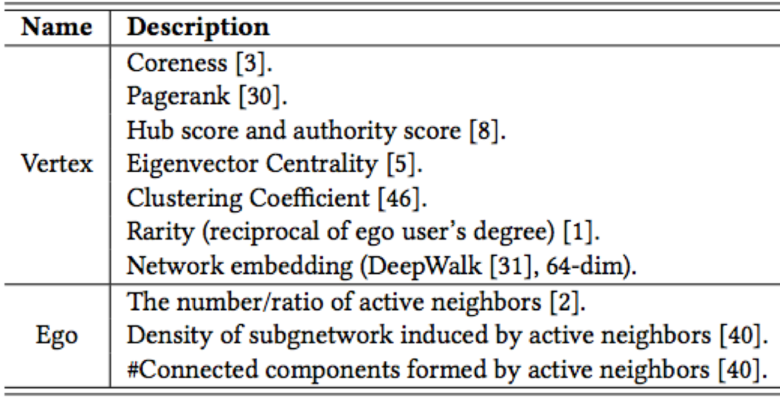


Given the five friends in red did something, whether the central users will do the same thing later, such as retweeting in Twitter, “like” in FB, or product purchase?

1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD’18.

Previous Solution

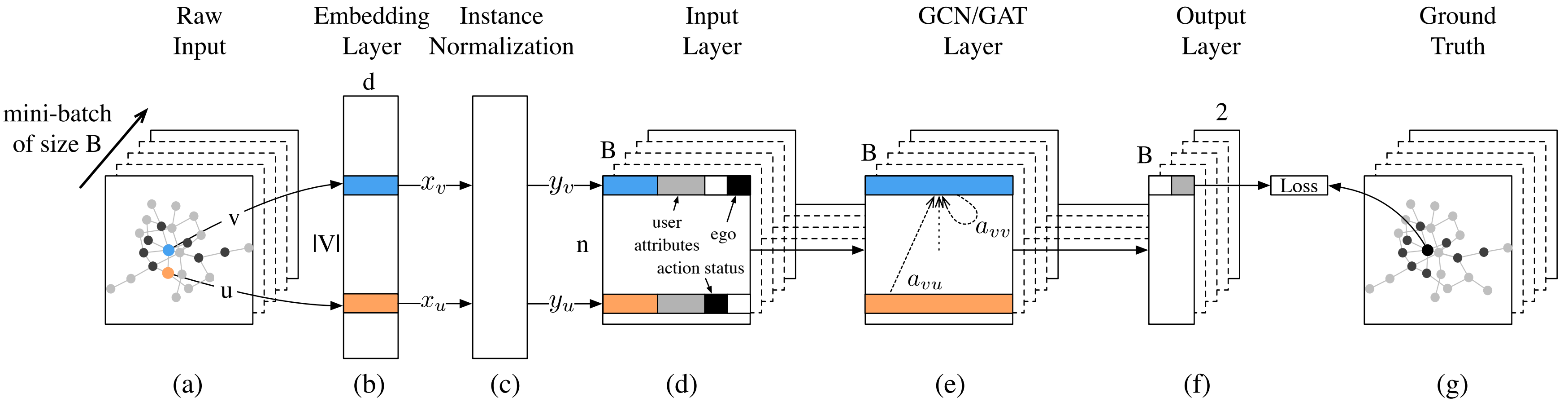




Name	Description
Vertex	Coreness [3].
	Pagerank [30].
	Hub score and authority score [8].
	Eigenvector Centrality [5].
	Clustering Coefficient [46].
	Rarity (reciprocal of ego user's degree) [1].
	Network embedding (DeepWalk [31], 64-dim).
Ego	The number/ratio of active neighbors [2].
	Density of subgnetwork induced by active neighbors [40].
	#Connected components formed by active neighbors [40].

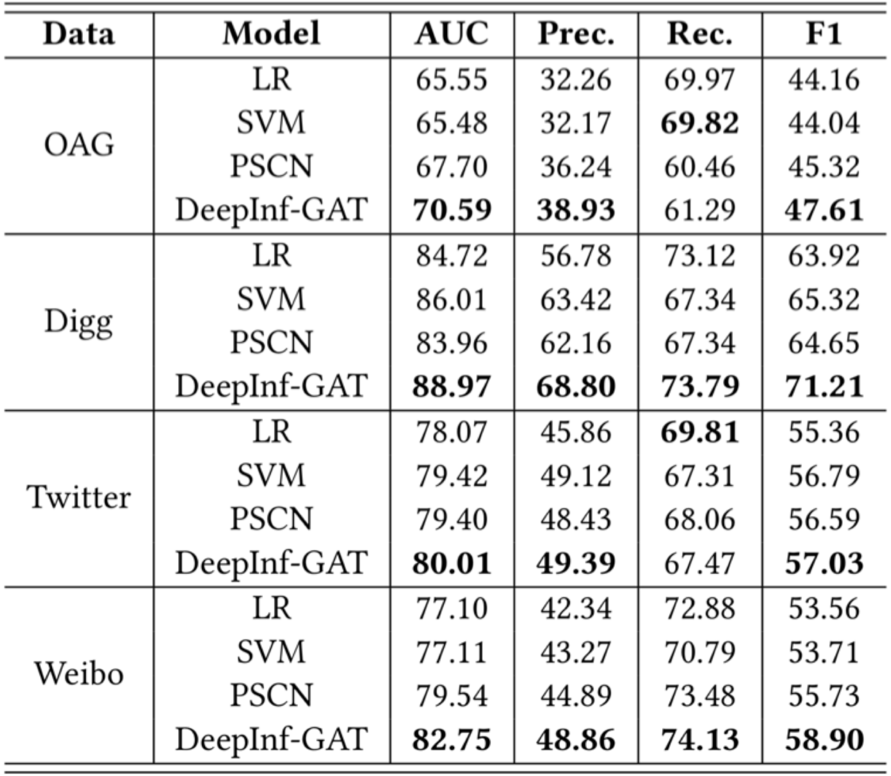
J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg. Structural diversity in social contagion. PNAS'12.

Graph Attention Networks



1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.

Experiments --- Results

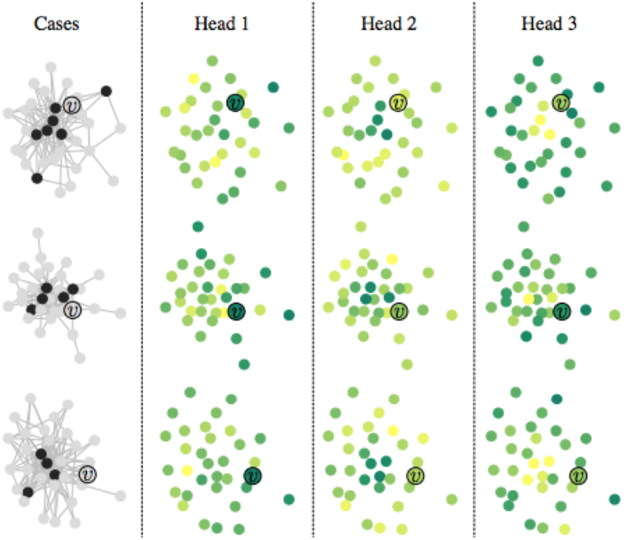


Data	Model	AUC	Prec.	Rec.	F1
OAG	LR	65.55	32.26	69.97	44.16
	SVM	65.48	32.17	**69.82**	44.04
	PSCN	67.70	36.24	60.46	45.32
	DeepInf-GAT	**70.59**	**38.93**	61.29	**47.61**
Digg	LR	84.72	56.78	73.12	63.92
	SVM	86.01	63.42	67.34	65.32
	PSCN	83.96	62.16	67.34	64.65
	DeepInf-GAT	**88.97**	**68.80**	**73.79**	**71.21**
Twitter	LR	78.07	45.86	**69.81**	55.36
	SVM	79.42	49.12	67.31	56.79
	PSCN	79.40	48.43	68.06	56.59
	DeepInf-GAT	**80.01**	**49.39**	67.47	**57.03**
Weibo	LR	77.10	42.34	72.88	53.56
	SVM	77.11	43.27	70.79	53.71
	PSCN	79.54	44.89	73.48	55.73
	DeepInf-GAT	**82.75**	**48.86**	**74.13**	**58.90**

1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.

Case Study

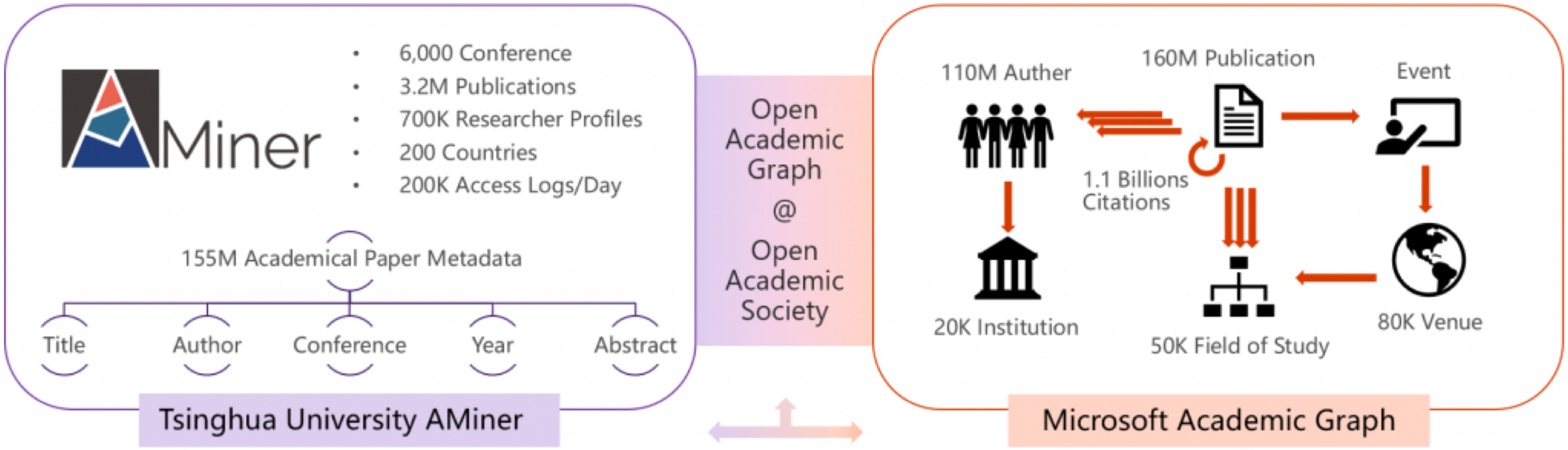
- How different graph attention heads highlight different areas of the network.
    - Head 1: Focus on the ego-user
    - Head 2: Highlight active users
    - Head 3: Highlight inactive users



1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.

LinKG: Knowledge graph linking with heterogeneous graph attention

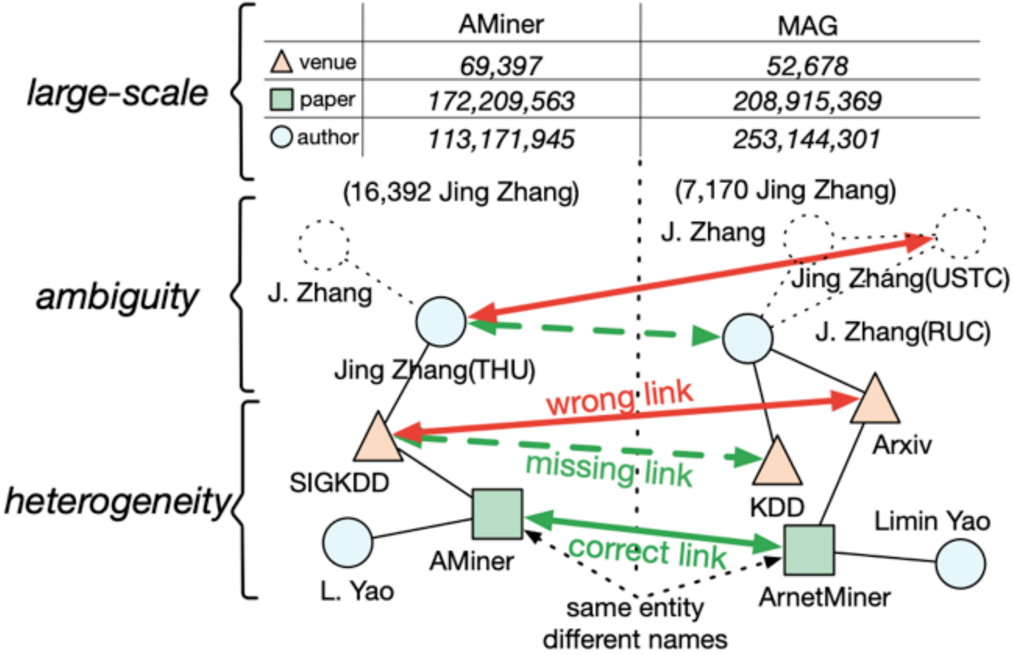
- **Input**: two heterogeneous entity graphs HG_1 and HG_2.
- **Output**: entity linkings $L = \{(e_1, e_2) | e_1 \in HG_1, e_2 \in HG_2\}$ such that e_1 and e_2 represent exactly the same entity.



---

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

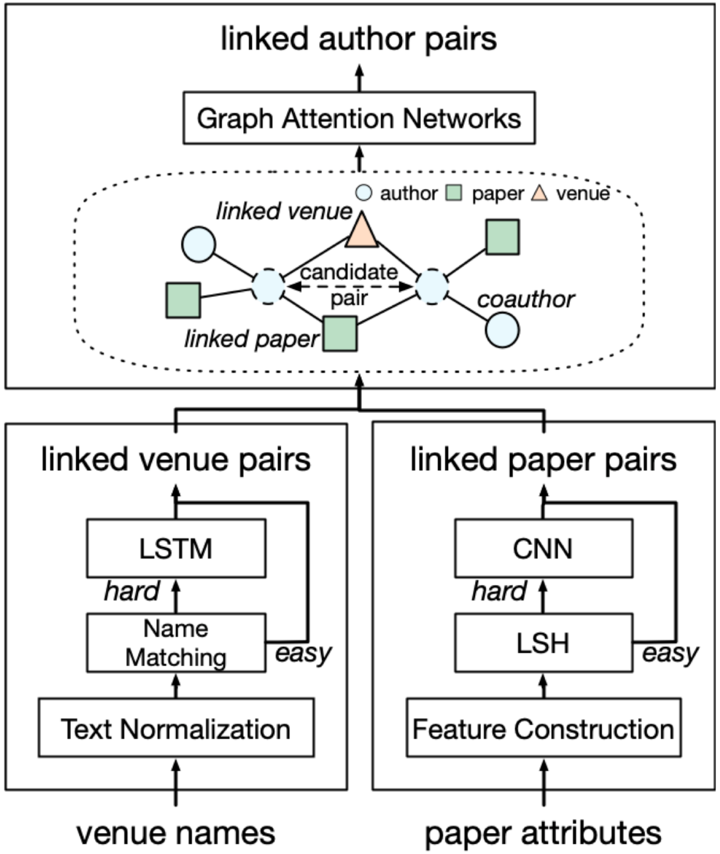
Linking large-scale heterogeneous academic graphs



	AMiner	MAG
△ venue	*69,397*	*52,678*
🟩 paper	*172,209,563*	*208,915,369*
○ author	*113,171,945*	*253,144,301*

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

Solution -- LinKG



Author linking module

Venue linking module

Paper linking module

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

Author linking model — Heterogenous Graph Attention

- Encoder layers
  - attention coefficient attn $(e_i\ , e_j)$ learnt by self-attention mechanism

$$o_{ij} = \mathrm{attn}(Wh_i, Wh_j)$$

  - Normalized attention coefficient: differentiate different types of entities

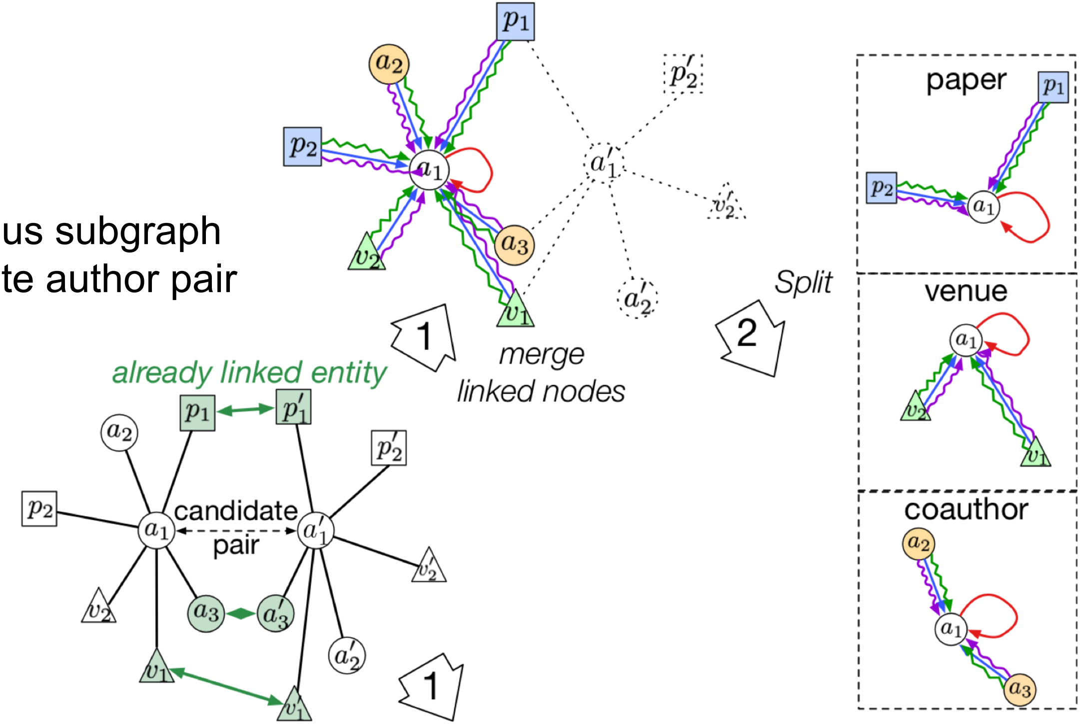
$$\alpha_{ij} = \frac{\exp(\mathrm{LeakyReLU}(c_{\tau(e_i)}^{\top} Wh_i + c_{\tau(e_j)}^{\top} Wh_j))}{\sum_{k \in \mathcal{N}_i} \exp(\mathrm{LeakyReLU}(c_{\tau(e_i)}^{\top} Wh_i + c_{\tau(e_k)}^{\top} Wh_k))}$$

aggregation weight of source entity e_j 's
embedding on target entity e_i

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

Author linking model — Heterogenous Graph Attention

Heterogeneous subgraph for a candidate author pair



Different attention parameters for different entity type

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(c^{\top}_{\tau(e_i)} W h_i + \boxed{c^{\top}_{\tau(e_j)}} W h_j))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(c^{\top}_{\tau(e_i)} W h_i + c^{\top}_{\tau(e_k)} W h_k))}$$

Attention coefficient

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

Experimental Results

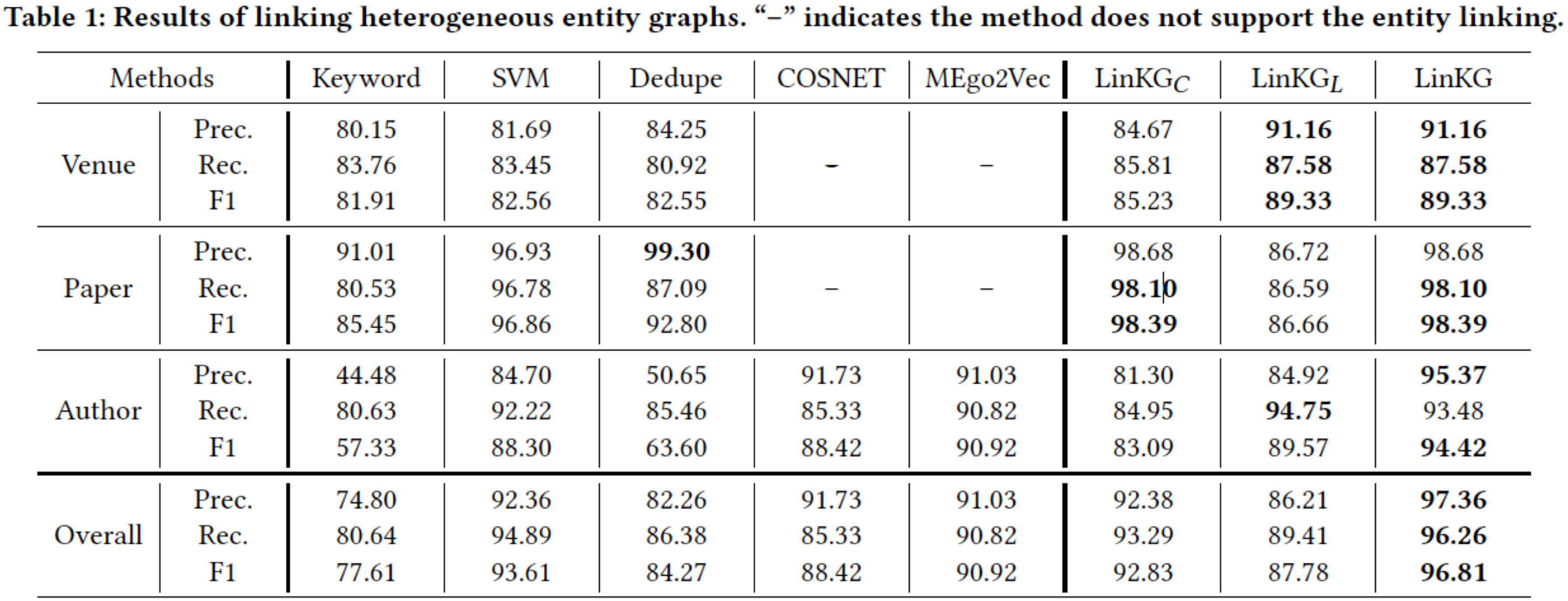


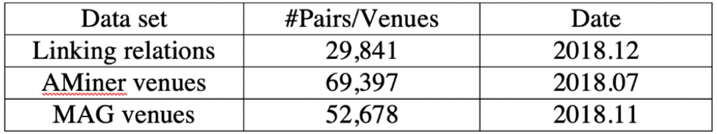
Table 1: Results of linking heterogeneous entity graphs. "–" indicates the method does not support the entity linking.

Methods		Keyword	SVM	Dedupe	COSNET	MEgo2Vec	LinKG$_C$	LinKG$_L$	LinKG
Venue	Prec.	80.15	81.69	84.25	–	–	84.67	**91.16**	**91.16**
	Rec.	83.76	83.45	80.92			85.81	**87.58**	**87.58**
	F1	81.91	82.56	82.55			85.23	**89.33**	**89.33**
Paper	Prec.	91.01	96.93	**99.30**	–	–	98.68	86.72	98.68
	Rec.	80.53	96.78	87.09			**98.10**	86.59	**98.10**
	F1	85.45	96.86	92.80			**98.39**	86.66	**98.39**
Author	Prec.	44.48	84.70	50.65	91.73	91.03	81.30	84.92	**95.37**
	Rec.	80.63	92.22	85.46	85.33	90.82	84.95	**94.75**	93.48
	F1	57.33	88.30	63.60	88.42	90.92	83.09	89.57	**94.42**
Overall	Prec.	74.80	92.36	82.26	91.73	91.03	92.38	86.21	**97.36**
	Rec.	80.64	94.89	86.38	85.33	90.82	93.29	89.41	**96.26**
	F1	77.61	93.61	84.27	88.42	90.92	92.83	87.78	**96.81**

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

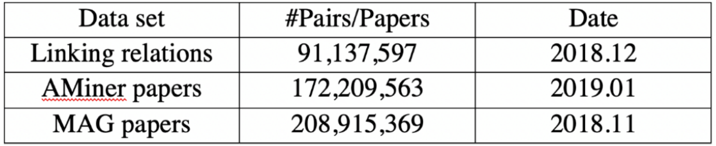
OAG: Open Academic Graph

https://www.openacademic.ai/oag/



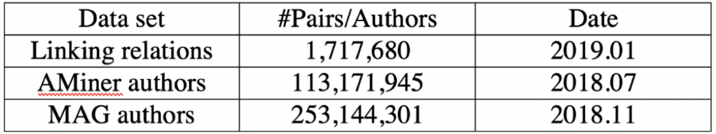
Data set	#Pairs/Venues	Date
Linking relations	29,841	2018.12
AMiner venues	69,397	2018.07
MAG venues	52,678	2018.11

Table 1: statistics of OAG venue data

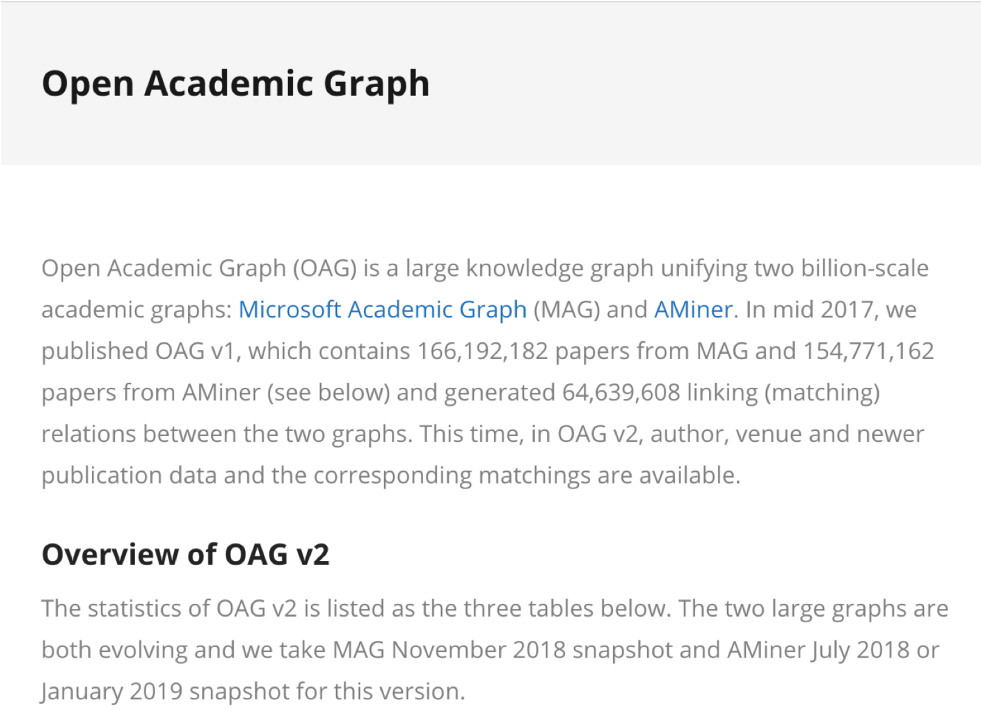


Data set	#Pairs/Papers	Date
Linking relations	91,137,597	2018.12
AMiner papers	172,209,563	2019.01
MAG papers	208,915,369	2018.11

Table 2: statistics of OAG paper data



Data set	#Pairs/Authors	Date
Linking relations	1,717,680	2019.01
AMiner authors	113,171,945	2018.07
MAG authors	253,144,301	2018.11



Open Academic Graph

Open Academic Graph (OAG) is a large knowledge graph unifying two billion-scale academic graphs: Microsoft Academic Graph (MAG) and AMiner. In mid 2017, we published OAG v1, which contains 166,192,182 papers from MAG and 154,771,162 papers from AMiner (see below) and generated 64,639,608 linking (matching) relations between the two graphs. This time, in OAG v2, author, venue and newer publication data and the corresponding matchings are available.

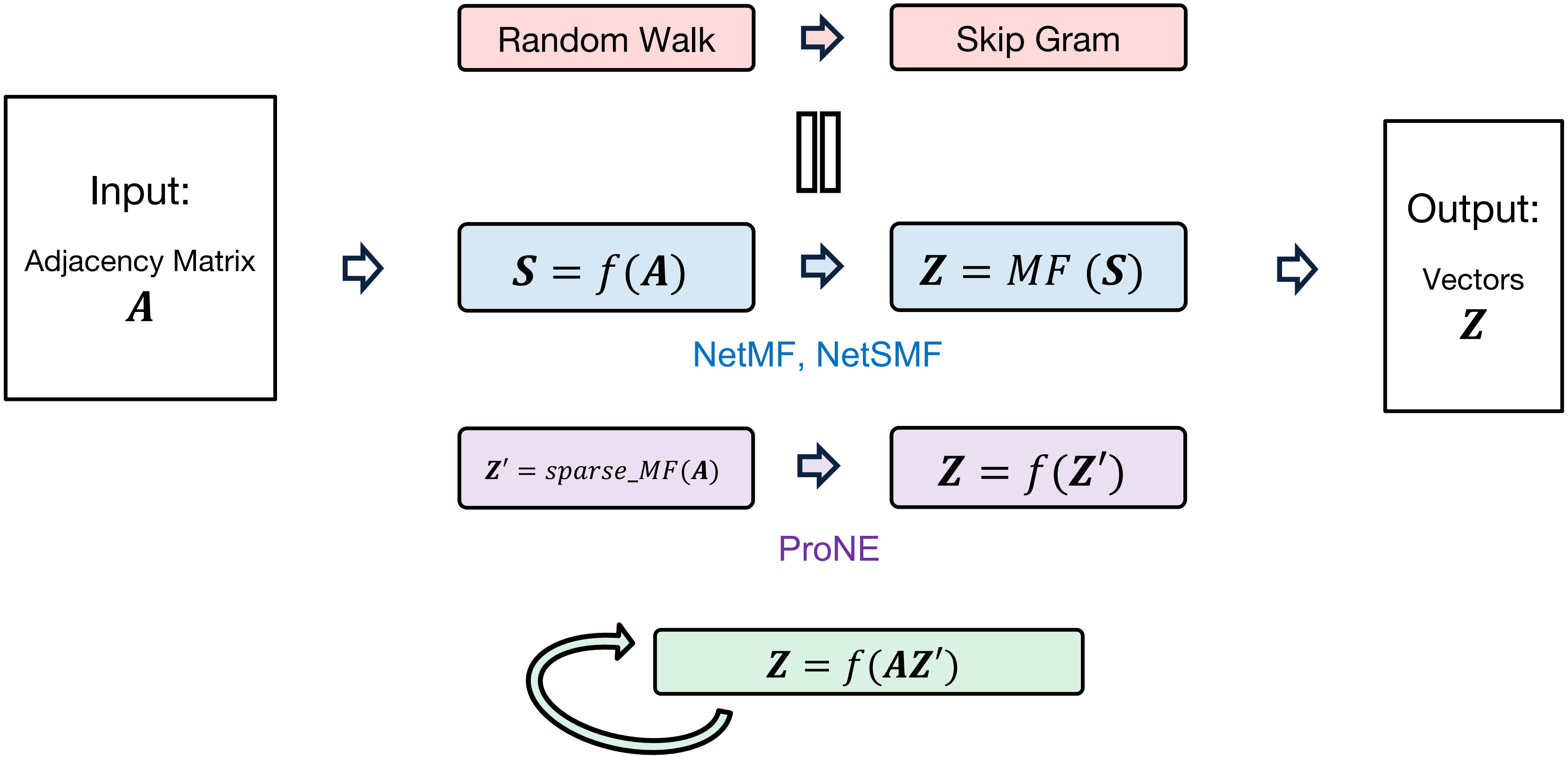
Overview of OAG v2

The statistics of OAG v2 is listed as the three tables below. The two large graphs are both evolving and we take MAG November 2018 snapshot and AMiner July 2018 or January 2019 snapshot for this version.

---

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

Connecting NE with graph neural networks



Input:

Adjacency Matrix

$\boldsymbol{A}$

Random Walk

Skip Gram

$\parallel$

$\boldsymbol{S} = f(\boldsymbol{A})$

$\boldsymbol{Z} = MF\ (\boldsymbol{S})$

NetMF, NetSMF

$\boldsymbol{Z}' = sparse\_MF(\boldsymbol{A})$

$\boldsymbol{Z} = f(\boldsymbol{Z}')$

ProNE

$\boldsymbol{Z} = f(\boldsymbol{A}\boldsymbol{Z}')$

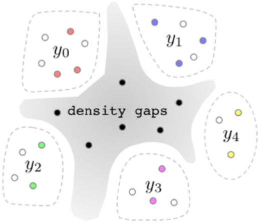
Output:

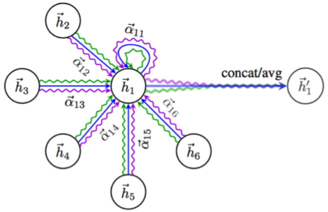
Vectors

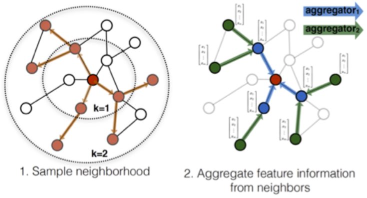
$\boldsymbol{Z}$

Graph neural networks

Graph Isomorphism Network, Deep Graph Infomax	2019: Velickovic et al. & Xu et al., ICLR’19
Graph attention	2018: Velickovic et al., ICLR’18
Neural message passing, GraphSage	2017: Gilmer et al., ICML’17; Hamilton et al., NIPS’17
Gated graph neural network	2016: Li et al., ICLR’16
structure2vec	2016: Dai et al., ICML’16
Graph convolutional network	2015: Duvenaud et al., NIPS’15; Kipf & Welling ICLR’17
Spectral graph convolution	2014: Bruna et al., ICLR’14
Graph neural network	2005: Gori et al., IJCNN’05

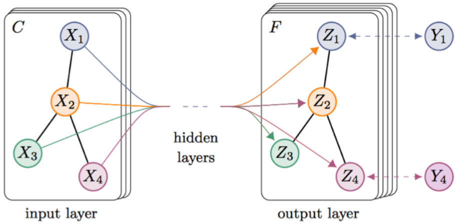






1. Sample neighborhood

2. Aggregate feature information from neighbors



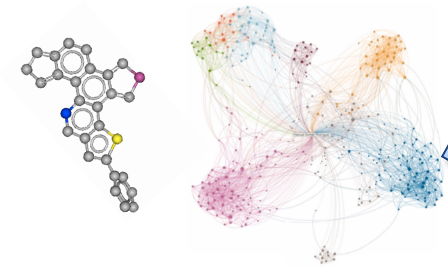
References

1. Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. WWW'19.
2. Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. WSDM’18.
3. Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. ProNE: Fast and Scalable Network Representation Learning. IJCAI'19.
4. Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. DeepInf: Modeling Influence Locality in Large Social Networks. KDD’18.
5. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.
6. Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
7. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017
8. Velickovic et al. Graph Attention Networks. ICLR 2018
9. Perozzi et al. DeepWalk: Online learning of social representations. In *KDD’ 14*.
10. Tang et al. LINE: Large scale information network embedding. In *WWW’15*.
11. Grover and Leskovec. node2vec: Scalable feature learning for networks. *In KDD’16.*
12. Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *KDD* 2017.

Graph Neural Networks:
A Learning Perspective

Learning from graph data

Networks



Timing

Feature/algorithm design

(city=Atlanta) AND (age=40)

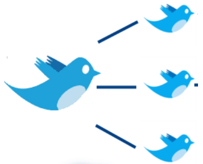
(neighbor=A) XOR (neighbor=B)

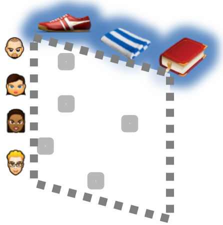
(bought=car) OR (time<3 years)

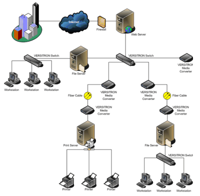
How to represent?

How to update?

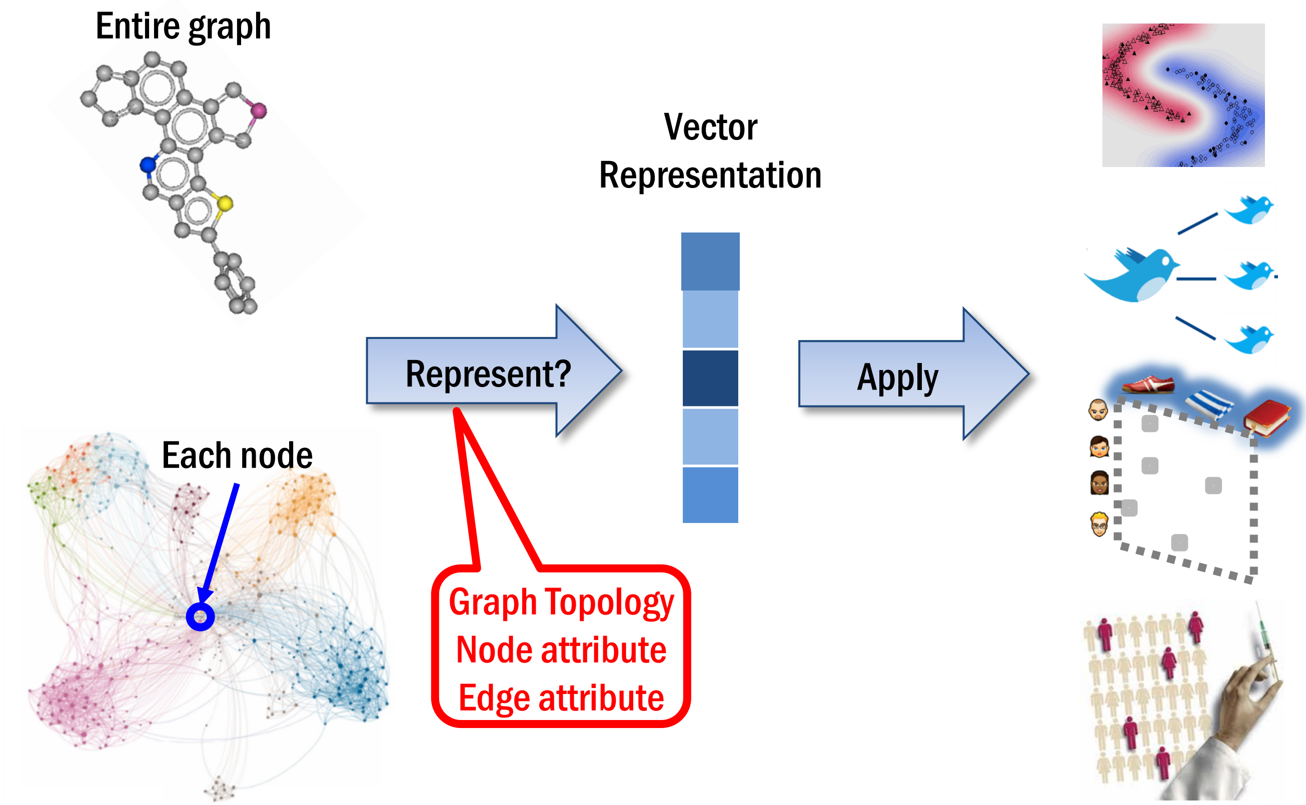
Various apps







Fundamental problem and challenge



Entire graph

Each node

Represent?

Graph Topology
Node attribute
Edge attribute

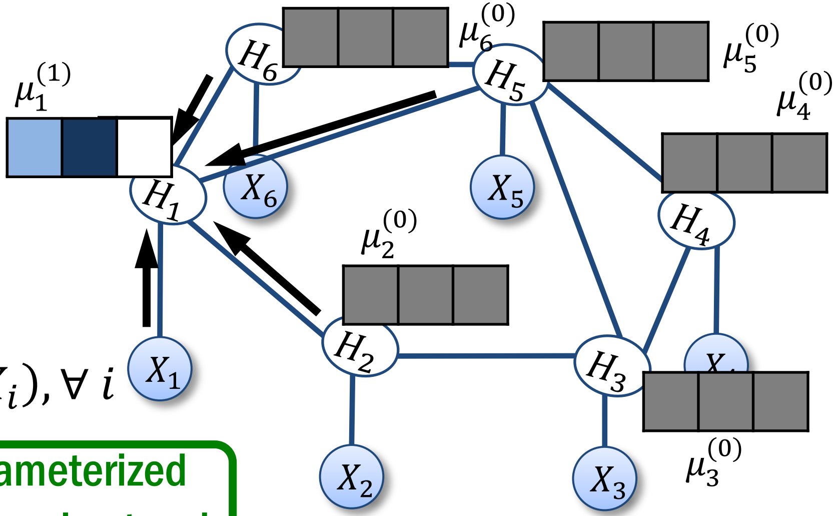
Vector
Representation

Apply

GCN/GNN/MPN/Structure2Vec

Obtain embedding via

iterative update algorithm:



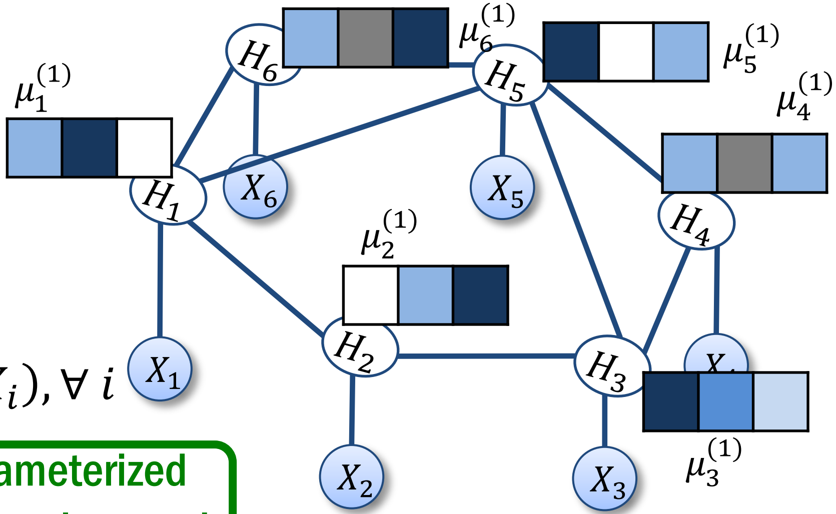
1. Initialize $\mu_i^{(0)} = \sigma(W_0 X_i), \forall\, i$
2. Iterate T times

Parameterized as neural network

$$\mu_i^{(t)} \leftarrow \sigma\left(W_1 \mu_i^{(t-1)} + W_2 \sum_{j \in \mathcal{N}(i)} \mu_j^{(t-1)}\right), \forall i$$

[Dai, et al. ICML 16]

GCN/GNN/MPN/Structure2Vec



Obtain embedding via

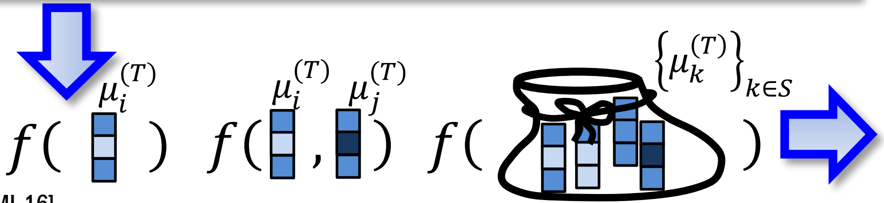
iterative update algorithm:

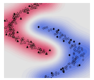
1. Initialize $\mu_i^{(0)} = \sigma(W_0 X_i), \forall\, i$

2. Iterate T times

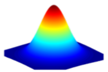
Parameterized as neural network

$$\mu_i^{(t)} \leftarrow \sigma\left(W_1 \mu_i^{(t-1)} + W_2 \sum_{j \in \mathcal{N}(i)} \mu_j^{(t-1)}\right), \forall i$$





Supervised Learning



Generative Models

Reinforcement Learning

Inductive: Generalize to New nodes & Graphs

[Dai, et al. ICML 16]

Variants of graph neural networks

Vanilla: $\mu_i^{(t)} \leftarrow \sigma\left(W_1 \mu_i^{(t-1)} + W_2 \sum_{j\in\mathcal{N}(i)} \mu_j^{(t-1)}\right)$

General: $\mu_i^{(t)} \leftarrow Update\left(\mu_i^{(t-1)}, Aggregate\left(\left\{\mu_j^{(t-1)}\right\}_{j\in S_1}, \left\{\mu_j^{(t-2)}\right\}_{j\in S_2}, \dots\right)\right)$

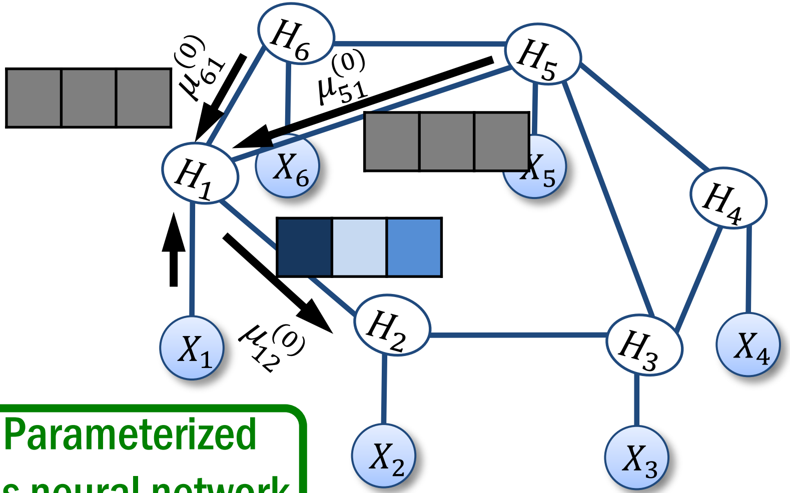
Residual: $\mu_i^{(t)} \leftarrow \mu_i^{(t-1)} + \sigma\left(W_1 \mu_i^{(t-1)} + \cdots\right)$

Gating: $\mu_i^{(t)} \leftarrow (1-\beta) \cdot \mu_i^{(t-1)} + \beta \cdot \sigma\left(W_1 \mu_i^{(t-1)} + \cdots\right)$

Attention: $\mu_i^{(t)} \leftarrow \sigma\left(\dots + W_2 \sum_{j\in\mathcal{N}(i)} \alpha_j \cdot \mu_j^{(t-1)}\right), \sum_{j\in\mathcal{N}(i)} \alpha_j = 1$

[Many papers ...]

Different message passing scheme



Obtain embedding via

iterative update algorithm:

1. Initialize $\mu_{ij}, \forall\ (i,j)$
2. Iterate T times

Parameterized as neural network

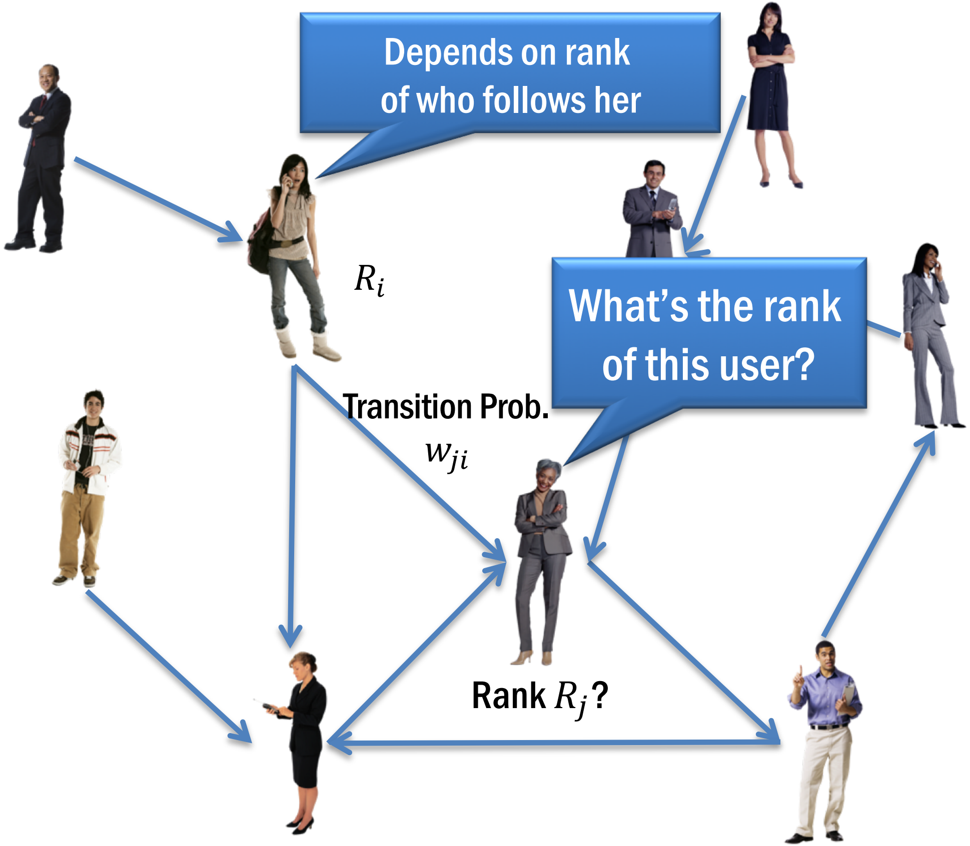
$$\boxed{\mu_{ij}^{(t)} \leftarrow \sigma\left(W_1 \mu_{ij}^{(t-1)} + W_2 \sum_{\ell \in \mathcal{N}(i) \setminus \mathrm{j}} \mu_{\ell i}^{(t-1)}\right), \forall (i,j)}$$

3. Aggregate $\mu_i = W_3 \sum_{\ell \in \mathcal{N}(i)} \mu_{\ell i}^{(T)}, \forall\ i$

[Dai, et al. ICML 16]

Connection to Other Graph Algorithms: More General

PageRank, specific feature



Depends on rank of who follows her

R_i

Transition Prob. w_{ji}

What's the rank of this user?

Rank R_j?

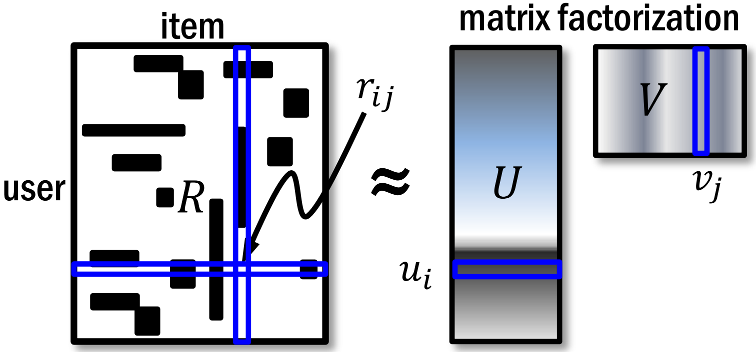
Iterate until convergence:

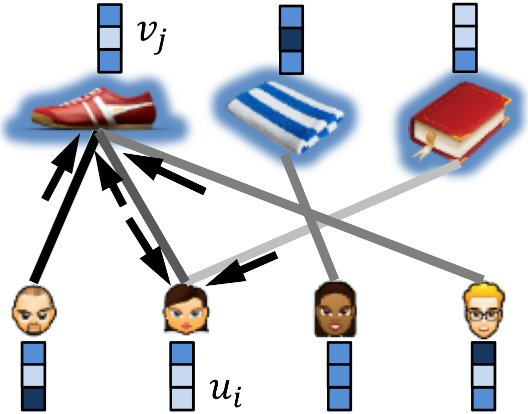
$$R_j \leftarrow \sum_{(j,i)\in E} w_{ji} R_i$$

Fixed update
Extract specifics

Loops in graph → Must iterate!

Matrix factorization, specific feature

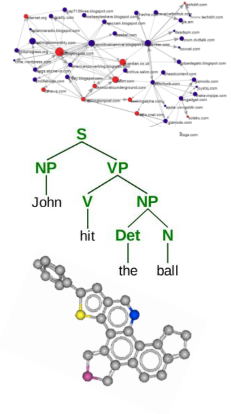




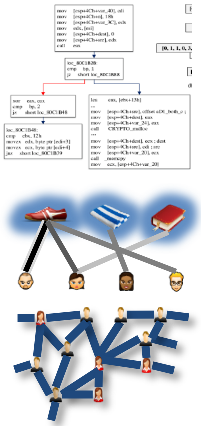
Alternating least square $||R - UV||_F^2$

```
1. Initialize $u_i, v_j, \forall i, j$
2. Iterate $T$ times
   • Update user factors:
     $u_i \leftarrow \text{argmin}_u \sum_{(j,i)\in E} \left(r_{ij} - u \cdot v_j\right)^2, \forall i$
   • Update item factors:
     $v_j \leftarrow \text{argmin}_v \sum_{(i,j)\in E} \left(r_{ij} - u_i \cdot v\right)^2, \forall j$
```

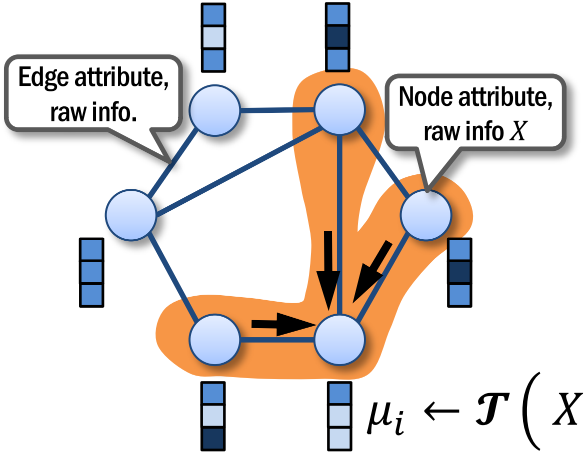
Fixed update
Extract specifics



GNN = Parametrized graph algorithm

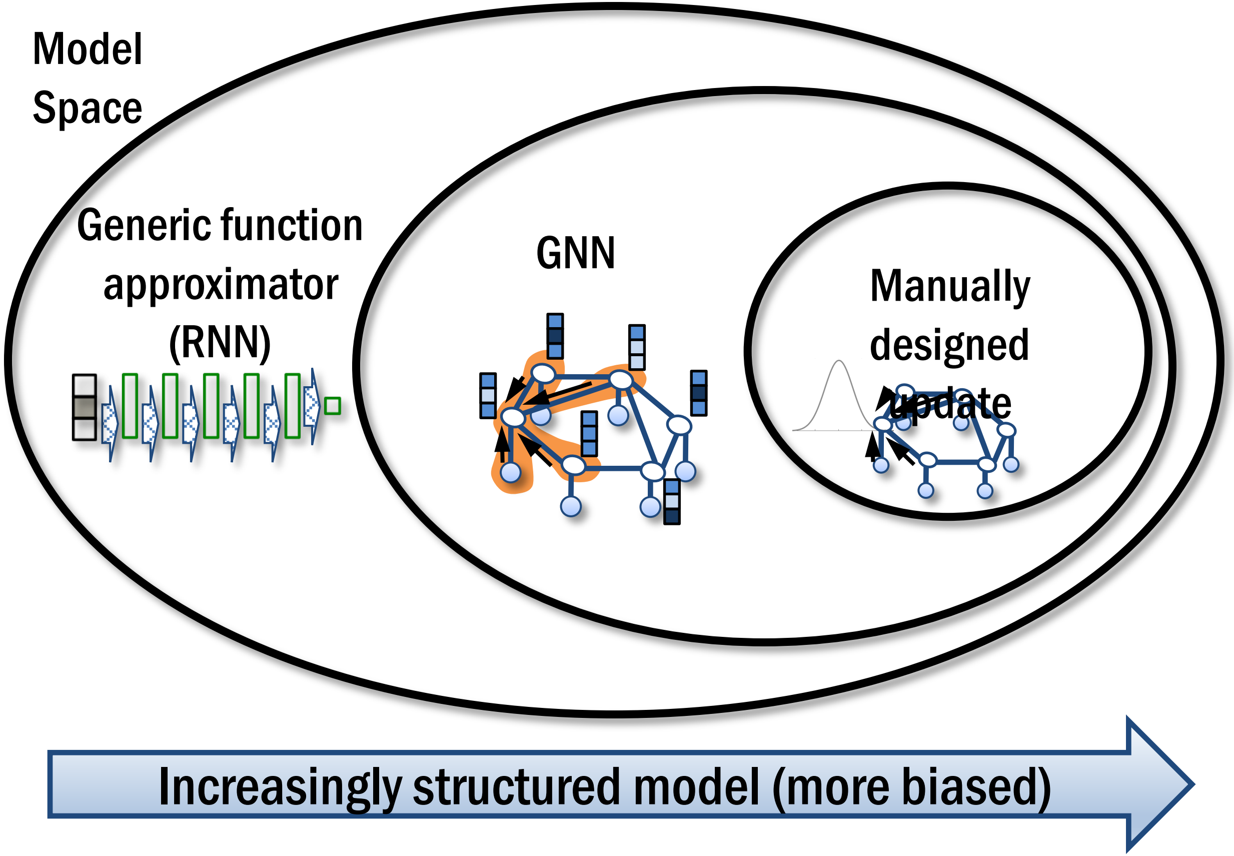


Graph Algorithm =
Graph Representation + Iterative Update



$$\mu_i \leftarrow \mathcal{T}\left(X_i, \{\mu_j\}_{j \in \mathcal{N}(i)} \right)$$

GNN is high structured deep model



Model Space

Generic function approximator (RNN)

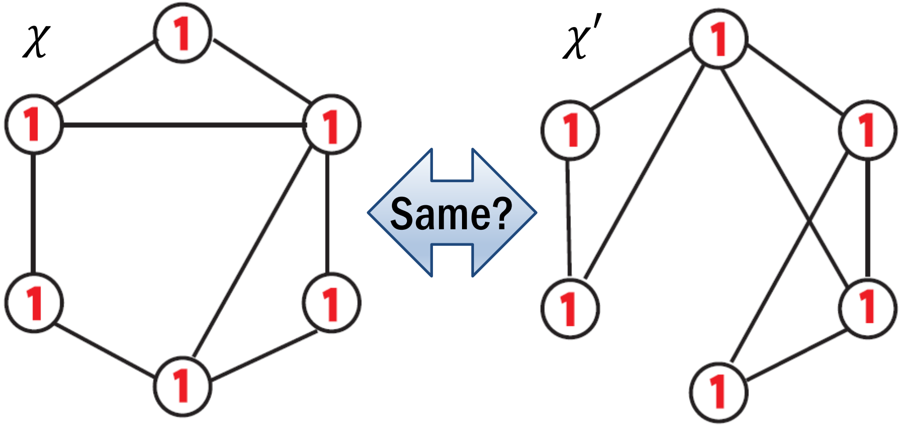
GNN

Manually designed update

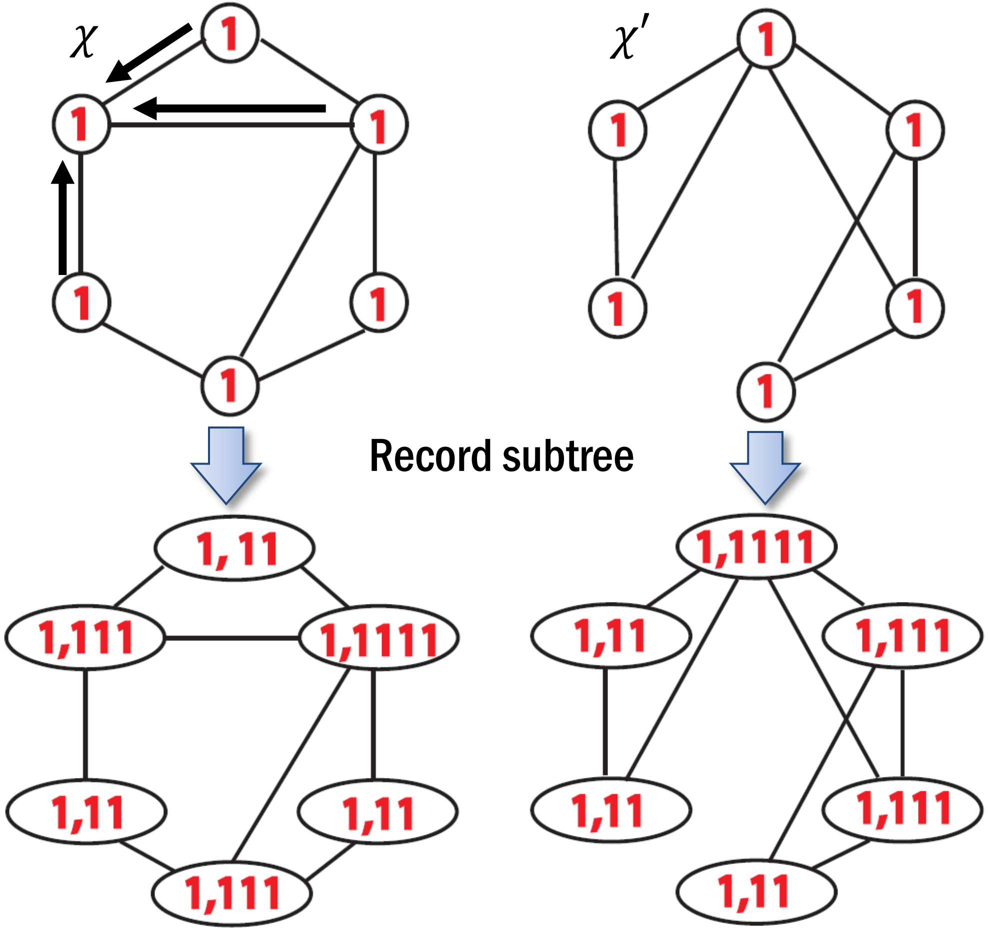
Increasingly structured model (more biased)

Connection to Graph Isomorphism:
Can Represent

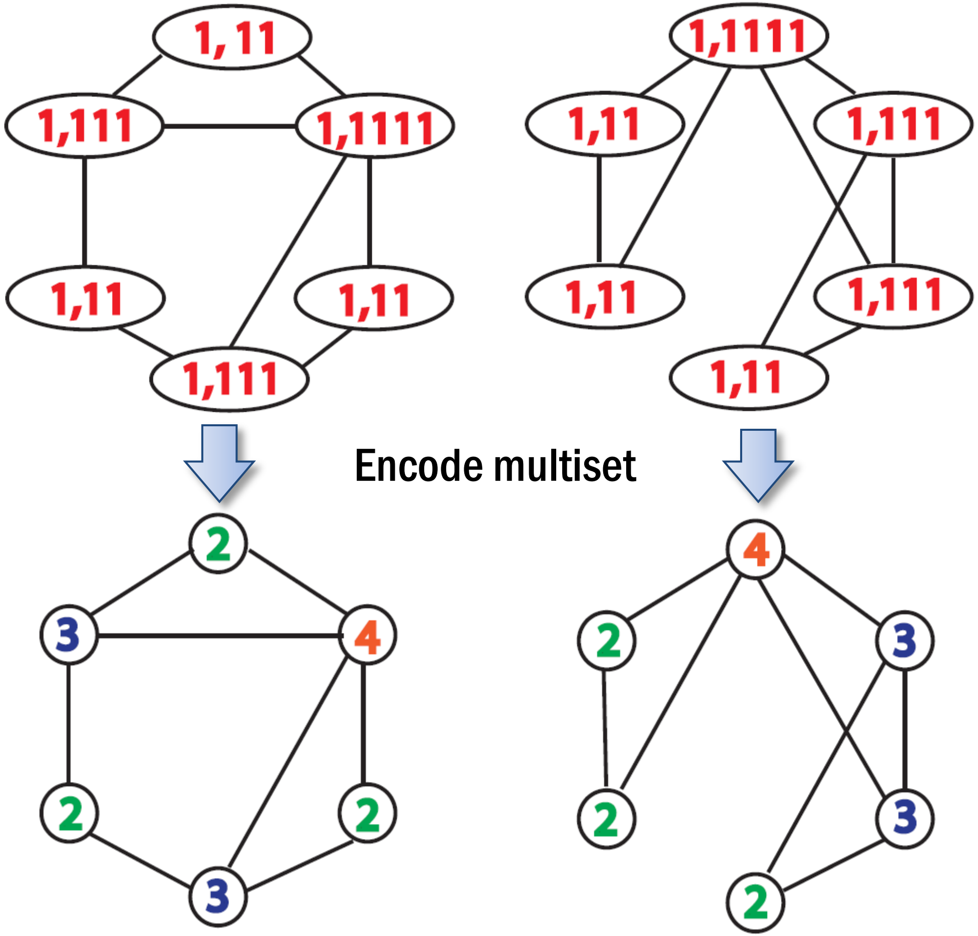
Graph isomorphism test



Weisfeiler-Lehmann algorithm: record subtree as multiset



Weisfeiler-Lehmann algorithm: encode (or hash)



Encode multiset

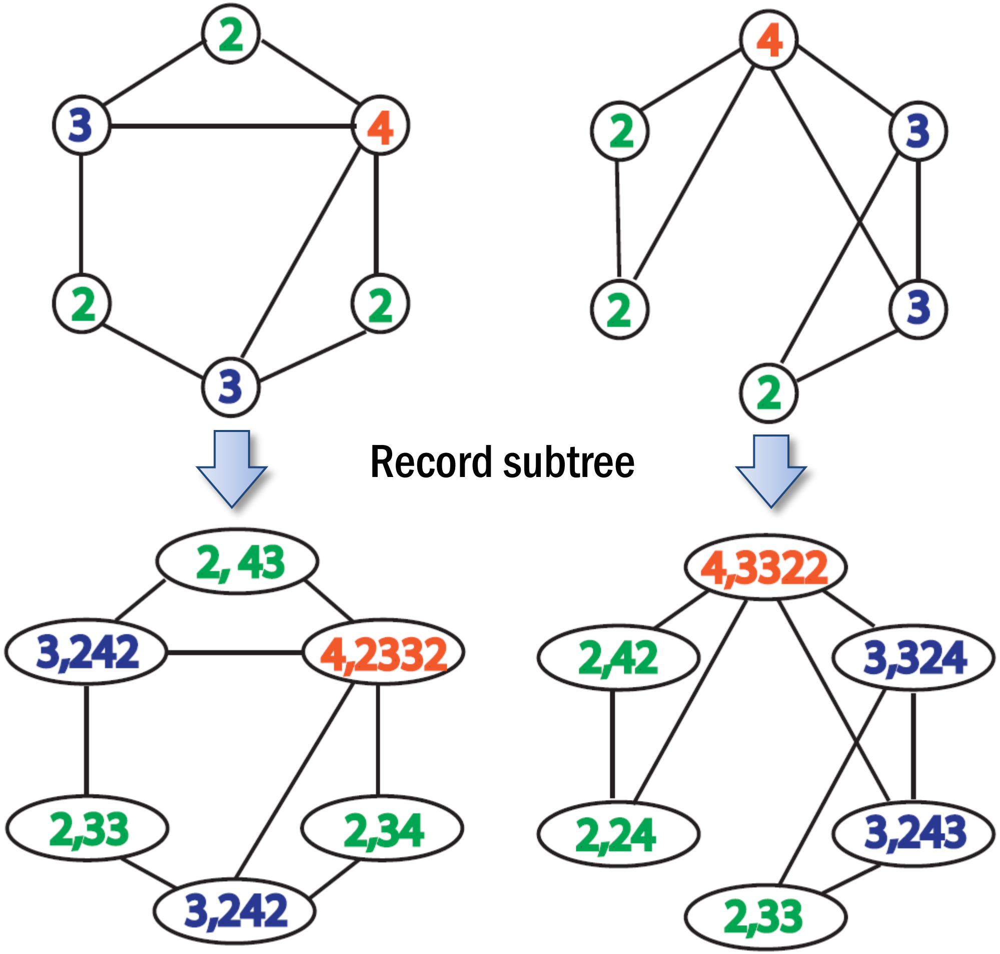
Map multisets to new codes

1, 11 ⟶ 2

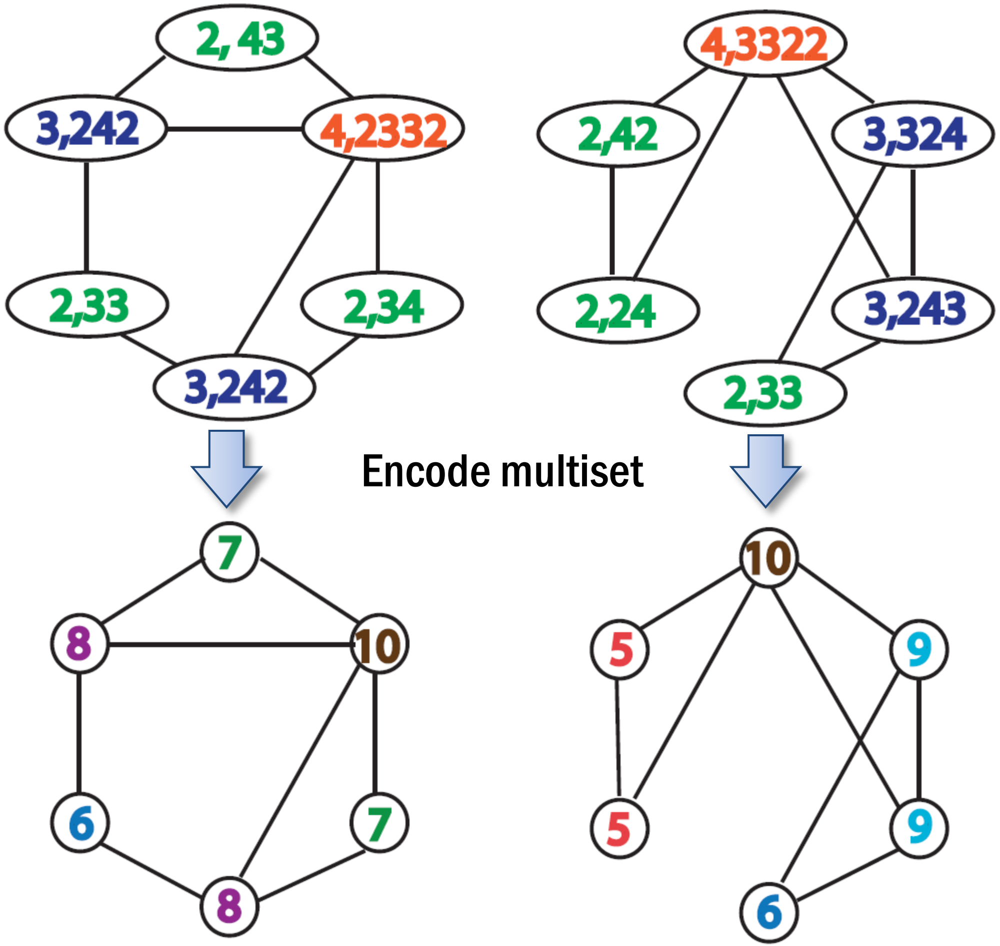
1,111 ⟶ 3

1,1111 ⟶ 4

Weisfeiler-Lehmann algorithm: record subtree as multiset again



Weisfeiler-Lehmann algorithm: encode (or hash) again

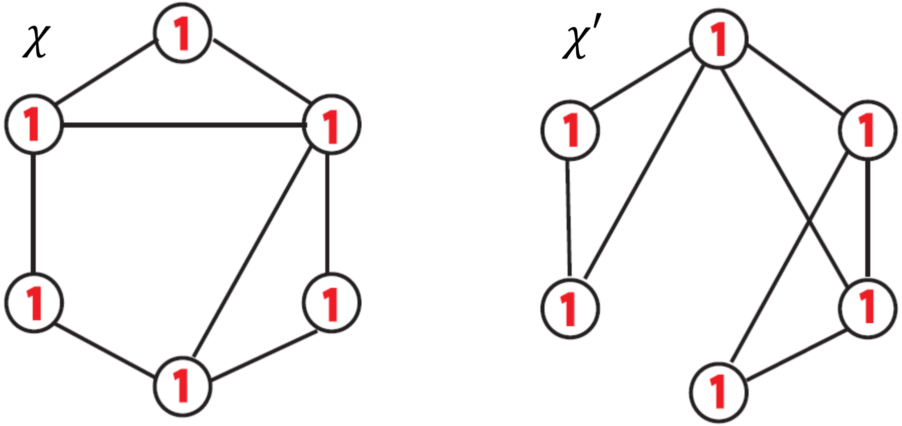


Encode multiset

Map multisets to new codes

2,24 → 5
2,33 → 6
2,34 → 7
3,224 → 8
3,234 → 9
4,2233 → 10

Weisfeiler-Lehmann algorithm: representation after T iterations



1 2 3 4 5 6 7 8 9 10 … … … …

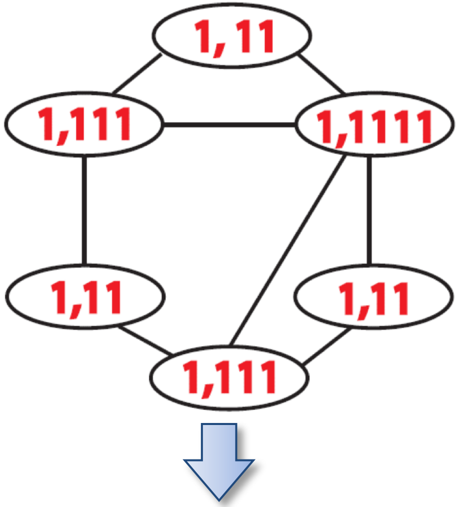
$$\phi(\chi) = (6, 3, 2, 1, 0, 1, 2, 2, 0, 1, \ldots\ldots\ldots\ldots)$$

$$\phi(\chi') = (\underset{\text{Level 0 feature}}{\underset{\uparrow}{6}}, \underbrace{3, 2, 1}_{\text{Level 1 feature}}, \underbrace{2, 1, 0, 0, 2, 1}_{\text{Level 2 feature}}, \underbrace{\ldots\ldots\ldots\ldots}_{\text{Level T features}})$$

Approximate Check!

1. If $\phi(\chi) \neq \phi(\chi')$, graphs not the same
2. Otherwise same graph or can not tell yet

Representation for multiset function



Map multisets

to new codes

1, 11 ⟶ 2

1,111 ⟶ 3

1,1111 ⟶ 4

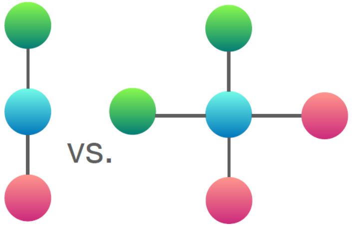
Vocabulary A, multiset $S \subset A$

1. **Representation:** $\forall g$ multiset function
$$g(S) = \gamma\left(\sum_{x \in S} f(x)\right)$$
with f a vector function

2. **One-to-one:** $\exists f$ s.t. $\sum_{x \in S} f(x)$ is unique for each finite multiset S

Key update: $\sigma\left(W_1 \mu_i^{(t)} + W_2 \sum_{j \in \mathcal{N}(i)} \mu_j^{(t)}\right)$

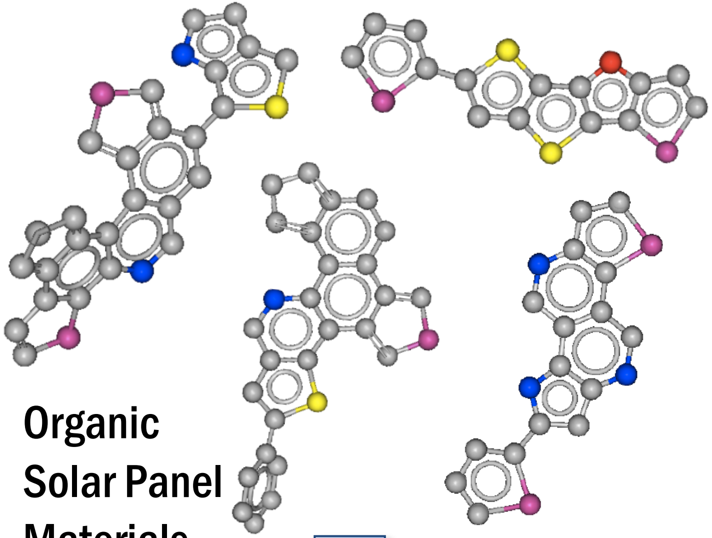
Average or max-pooling not as expressive



[Dai et al. 2016, Zaheer et al. 2017, Xu et al. 2019]

Benefit of GNN for Graph Feature Extraction Algorithm

Materials/Drug design



Organic Solar Panel Materials

predict

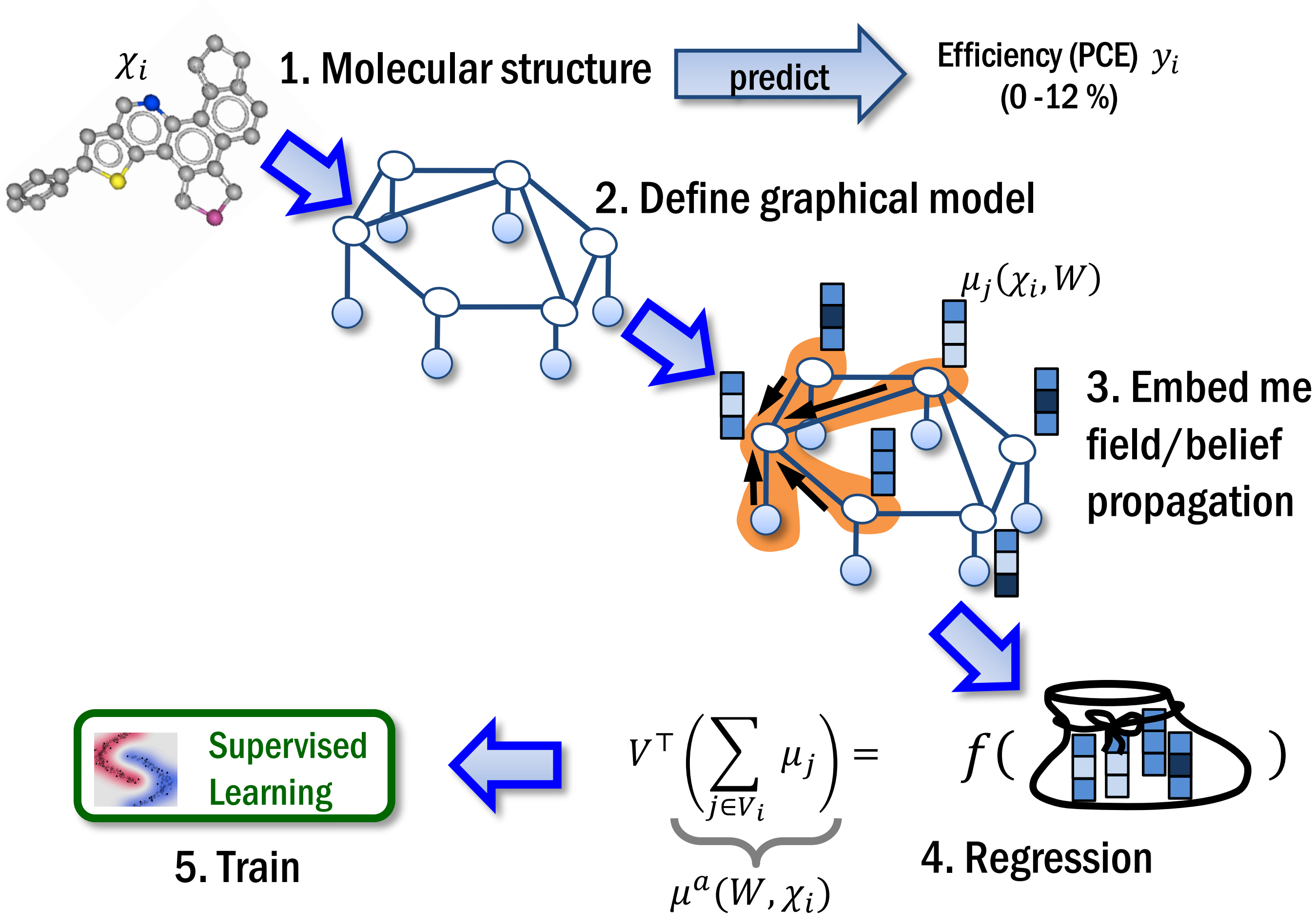
Power Conversion Efficiency (PCE) (0 -12 %)

Dataset	Harvard clean energy project
Size	2.3 million
Type	Molecule
Alphabet #	6
Avg node #	28
Avg edge #	33

feature	dimension	MAE
Level 3	1.6 million	0.143
Level 6	1.3 billion	0.096

[Dai et al. ICML 2016]

Prediction for structured data



1. Molecular structure

Efficiency (PCE) y_i
(0 -12 %)

2. Define graphical model

3. Embed mean field/belief propagation

$$V^\top \underbrace{\left(\sum_{j\in V_i} \mu_j\right)}_{\mu^a(W,\chi_i)} = f(\quad)$$

4. Regression

Supervised Learning

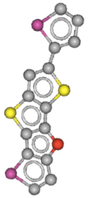
5. Train

Parameter learning

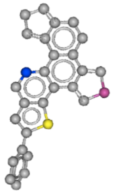
Given m data points $\{\chi_1, \chi_2, \dots, \chi_m\}$, estimate parameters W and V which minimize empirical loss

$$\min_{V,W} L(V,W) := \sum_{i=1}^{m} (y_i - V^\top \mu^a(W, \chi_i))^2$$

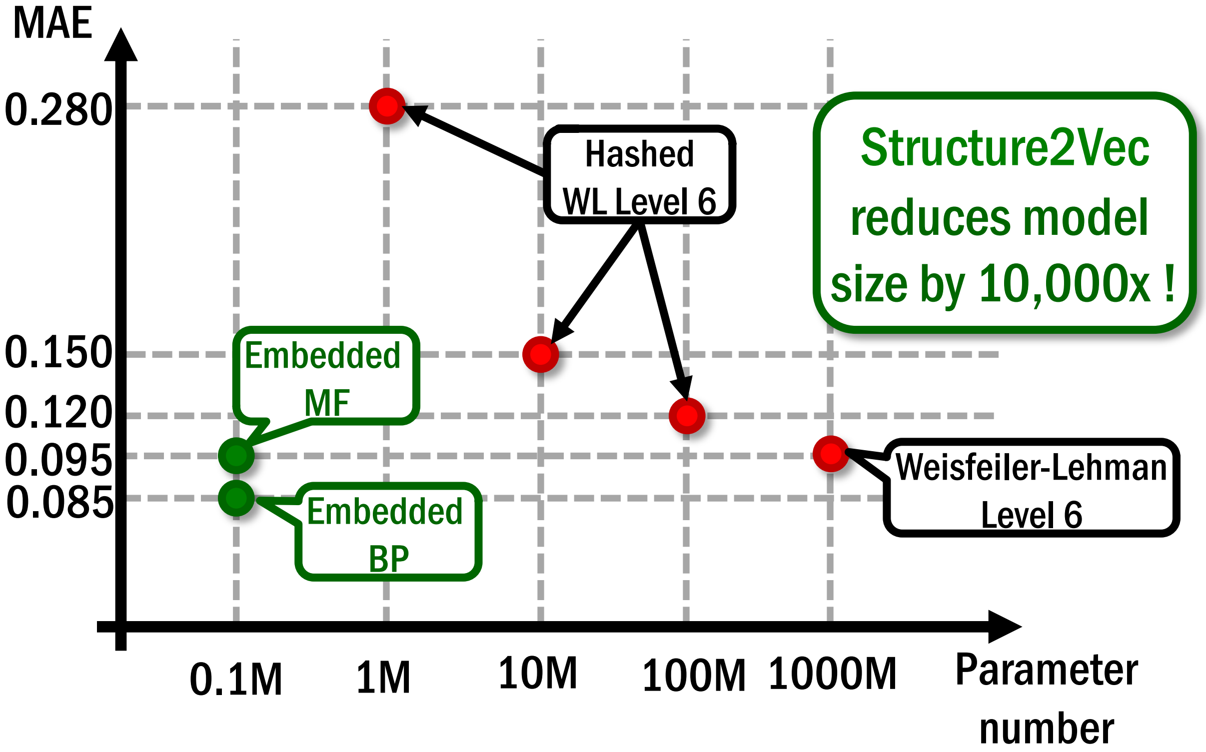
Computation	Operation	Similar to
Objective $L(V, W)$	A forward sequence of nonlinear mappings	Graphical model inference
Gradient $\frac{\partial L}{\partial W}$	Chain rule of derivatives in reverse order	Back propagation in deep learning



More compact model and lower error



Harvard clean energy dataset, 2.3 million organic molecules, predict power conversion efficiency (0 -12 %)

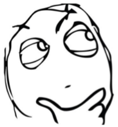


Structure2Vec reduces model size by 10,000x !

Fraudulent account detection

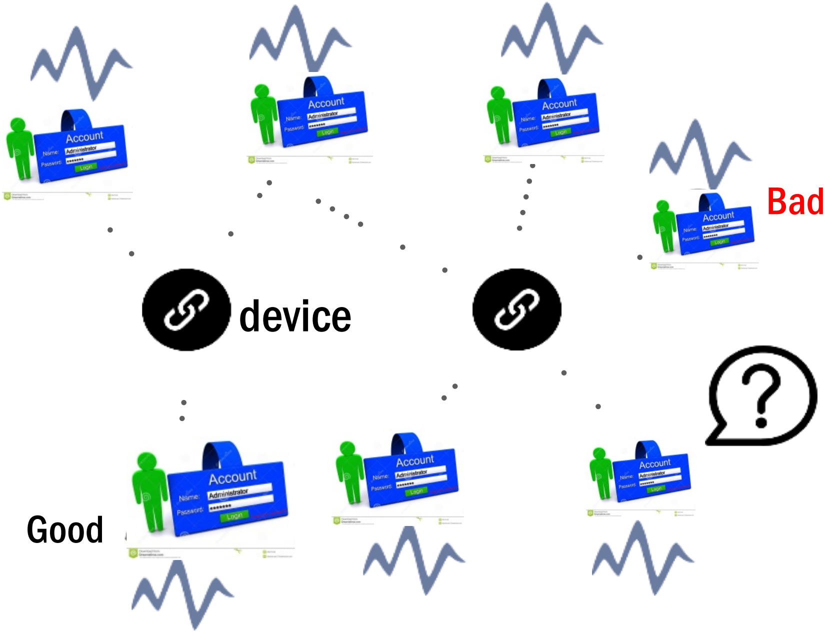
Alipay: new accounts in a month: millions of nodes and edges.

Fake account can increase system level risk



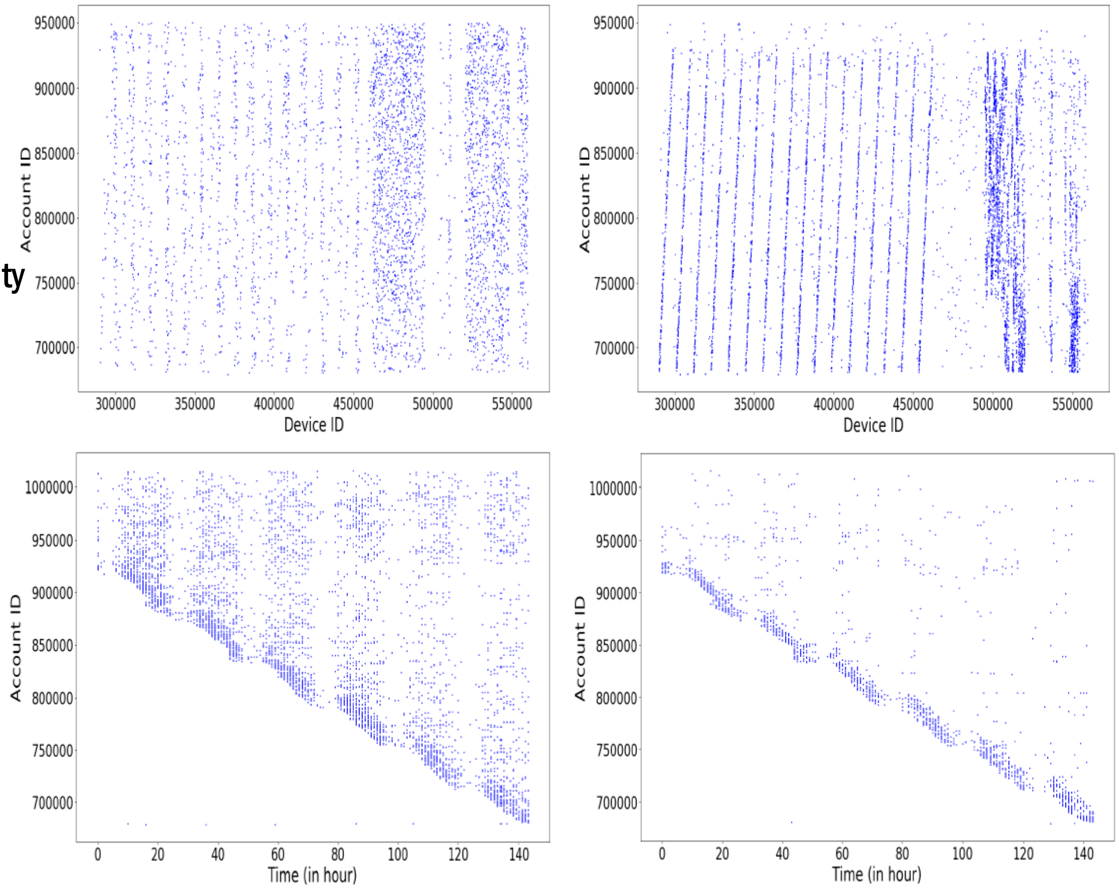
Leverage account activity + connectivity?

Account – Device Network



[Liu, et al. CCS 17, CIKM17]

Fraudulent account pattern



Normal account

Fake account

Device Connectivity

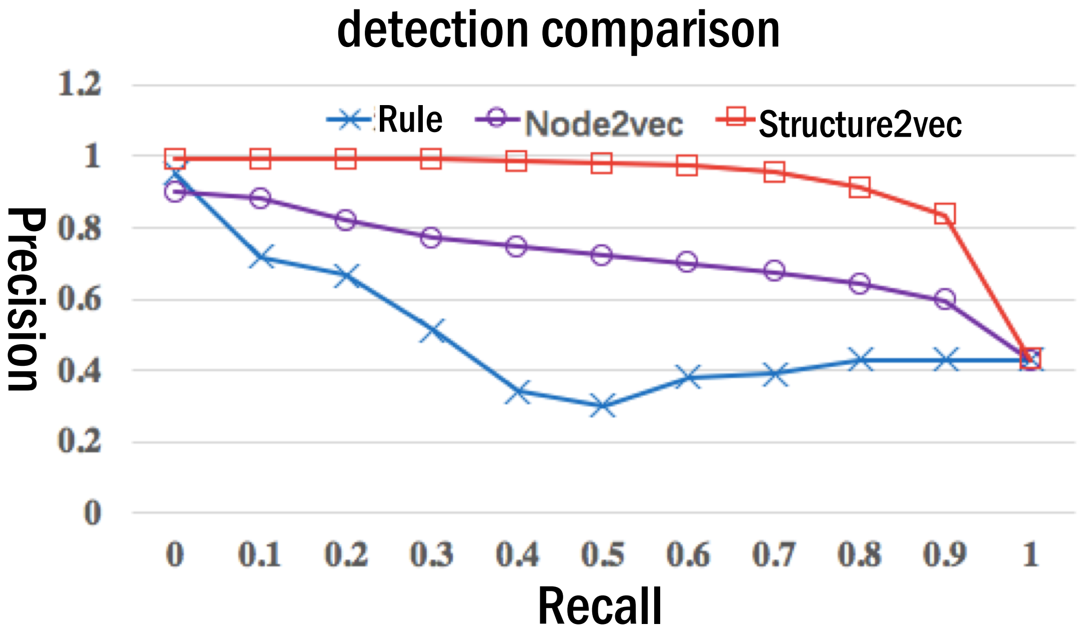
Account Activity

Results

< 1 percent of fraudulent accounts / month

High precision = less disturbance to user experience

High recall = detect more fraudulent account



GNN to Parametrize
Combinatorial Optimization Algorithm

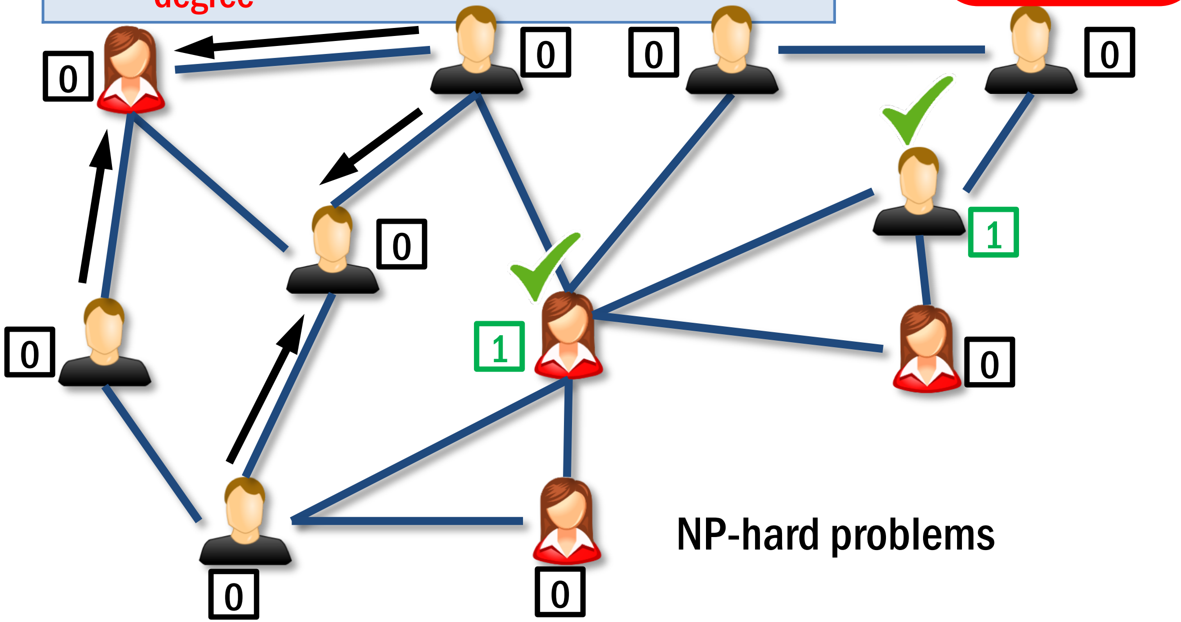
Combinatorial optimization over graph

2 - approximation for minimum vertex cover

Repeat till all edges covered:

- Select uncovered edge with largest total degree

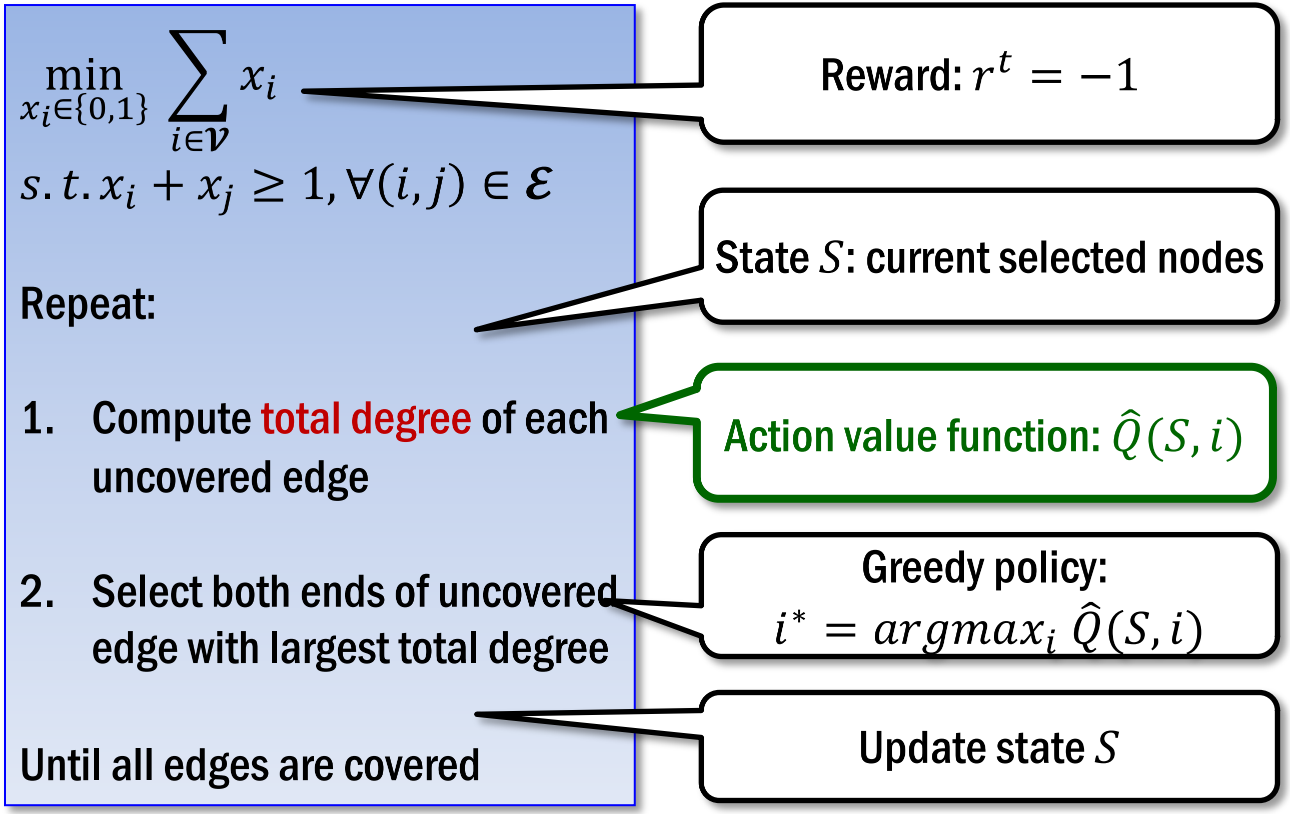
Manually designed rule. Can we learn from data?



NP-hard problems

Greedy algorithm as Markov decision process

Minimum vertex cover: smallest number of nodes to cover all edges



$$\min_{x_i \in \{0,1\}} \sum_{i \in \mathcal{V}} x_i$$

$$s.t. \, x_i + x_j \geq 1, \forall (i,j) \in \mathcal{E}$$

Repeat:

1. Compute total degree of each uncovered edge
2. Select both ends of uncovered edge with largest total degree

Until all edges are covered

Reward: $r^t = -1$

State S: current selected nodes

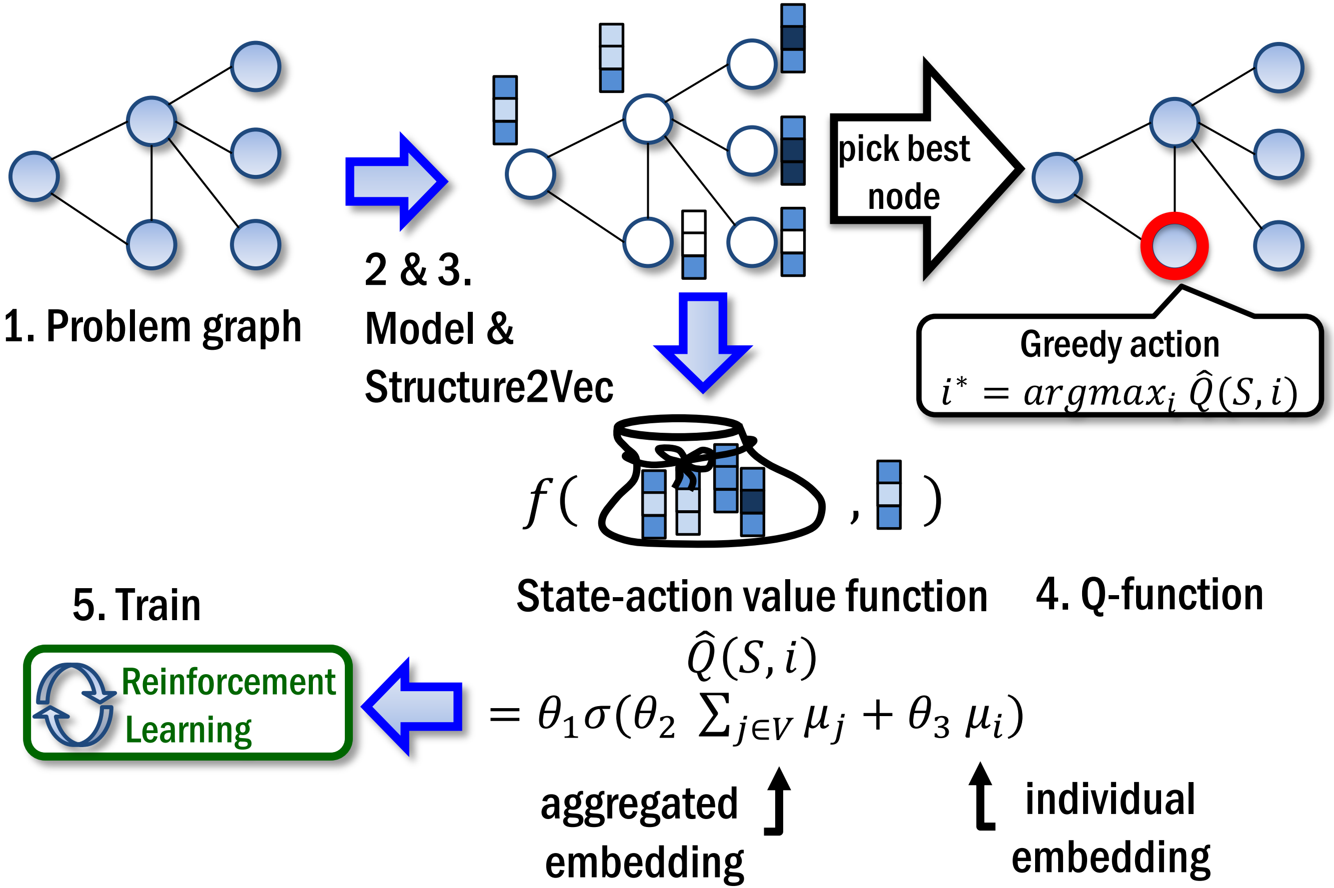
Action value function: $\hat{Q}(S,i)$

Greedy policy:
$i^* = argmax_i \, \hat{Q}(S,i)$

Update state S

[Dai et al. NIPS 2017]

Embedding for state-action value function



1. Problem graph

2 & 3. Model & Structure2Vec

pick best node

Greedy action

$i^* = argmax_i\ \hat{Q}(S,i)$

$f($, $)$

4. Q-function

State-action value function

$$\hat{Q}(S,i)$$
$$= \theta_1 \sigma(\theta_2\ \textstyle\sum_{j \in V} \mu_j + \theta_3\ \mu_i)$$

aggregated embedding

individual embedding

5. Train

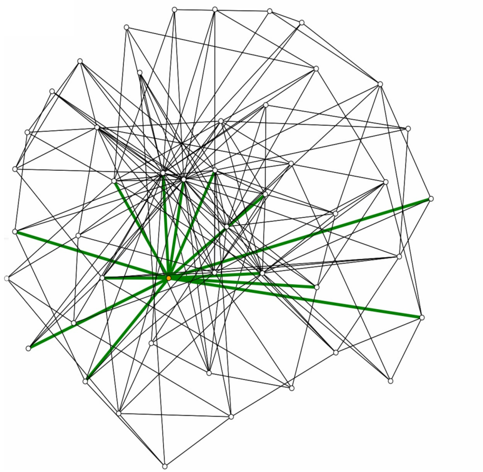
Reinforcement Learning

[Dai et al. NIPS 2017]

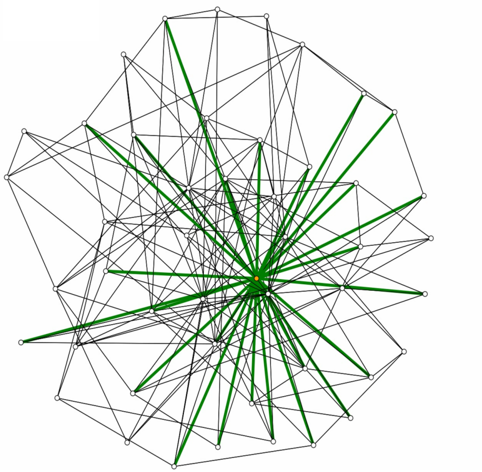
What new algorithm is learned?

Learned algorithm balances between

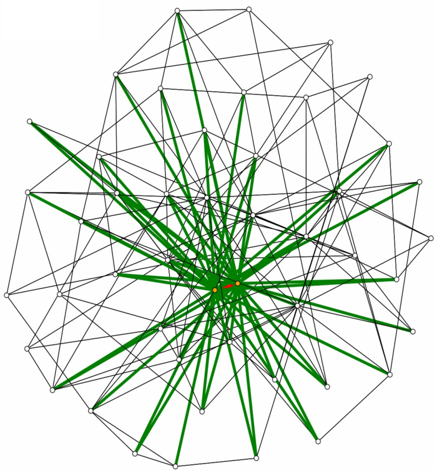
- degree of the picked node and
- fragmentation of the graph



Structure2Vec



Node greedy



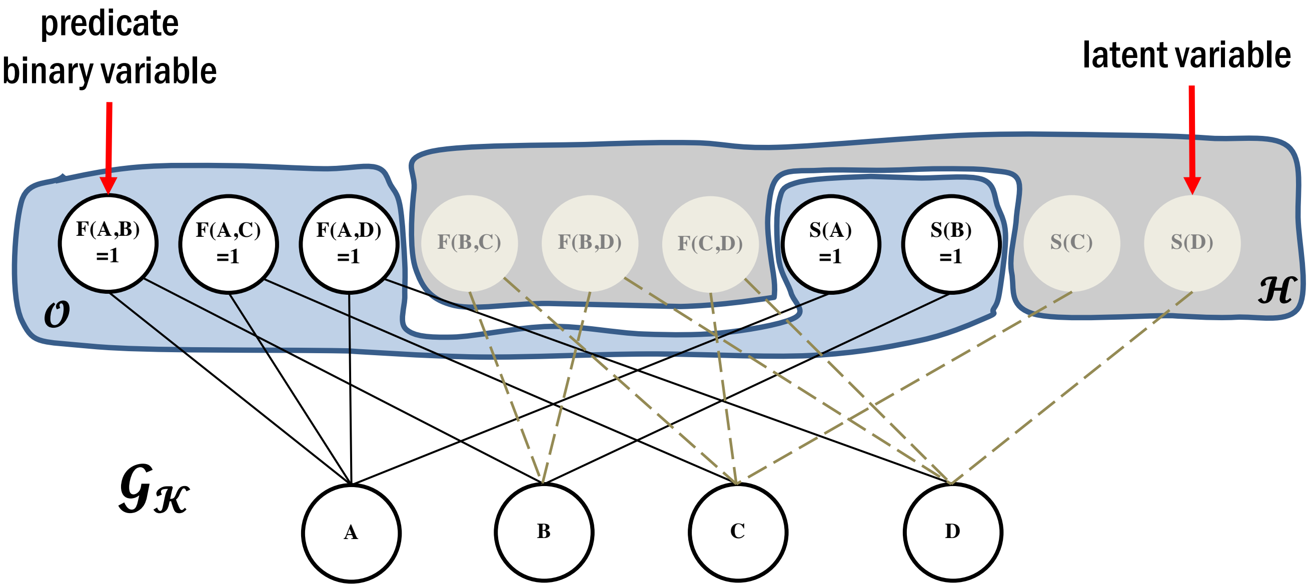
Edge greedy

[Dai et al. NIPS 2017]

GNN to Parametrize Variational Inference Algorithm for Probabilistic Logic

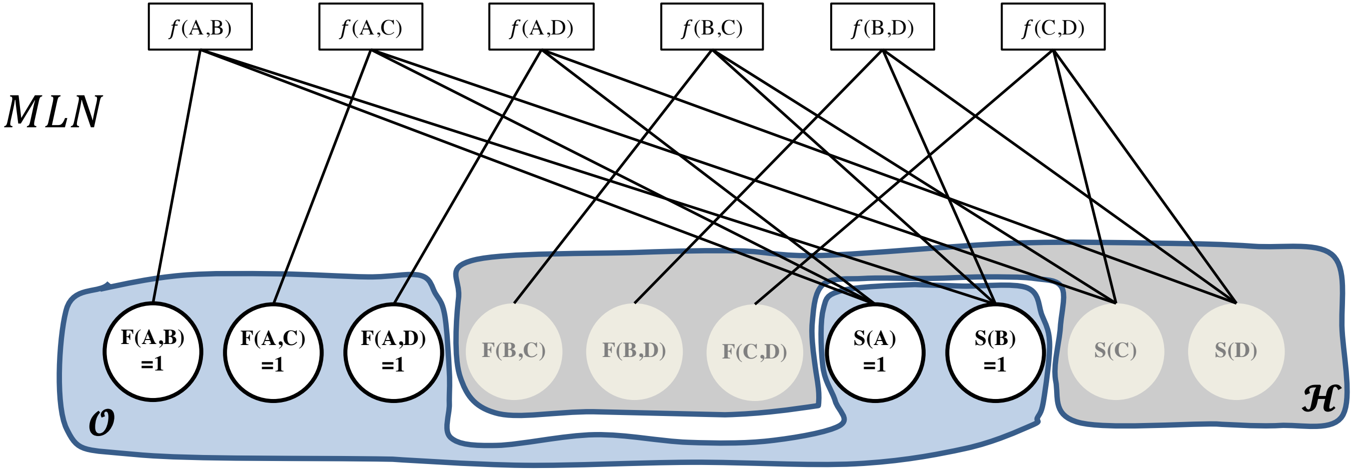
Factor graph representation for knowledge base

- Entity (constant), $\mathcal{C} = \{A, B, C, D \ldots\}$
- Predicate (attribute | relation), $r(\cdot): \mathcal{C} \times \mathcal{C} \times \cdots \mapsto \{0,1\}$
  - Eg. Smoke(x), Friend(x,x'), Like(x,x')



Markov logic networks

- Use logic formula $f(\cdot)$: $\mathcal{C}\times\mathcal{C}\times\cdots\times\mathcal{C}\mapsto\{0,1\}$ for potential functions
  - Eg. formula f(A,B): Friend(A,B) ∧ Smoke(A) ⇒Smoke(B)

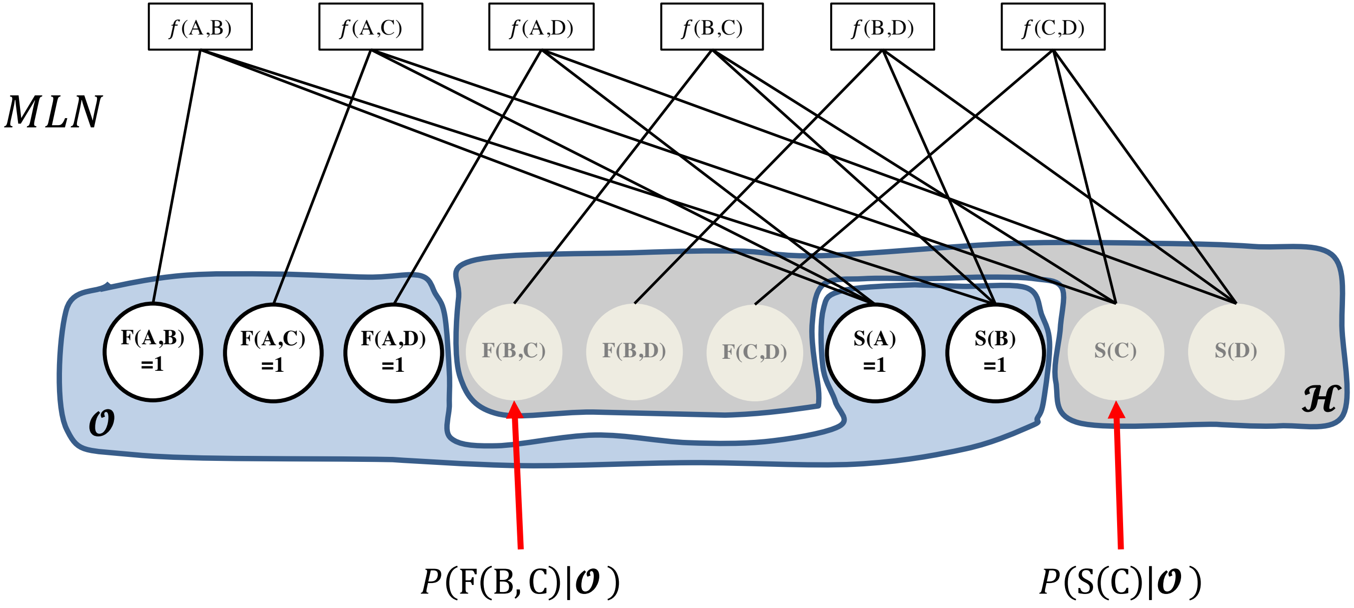


$$P(\mathcal{O},\mathcal{H})=\frac{1}{Z}\exp\left(\sum_{f}w_f\sum_{a_f}\phi_f(a_f)\right)$$

w_f: formula weight, $\phi_f(x,x')$: ¬F(x,x')∨¬S(x) ∨ S(x'), Z: normalization constant

Challenges in inference

- A large grounded network, $O(n^2)$ in the number n of entities!
- Enumerate configuration over $O(n^2)$ binary variables, with $O\left(2^{n^2}\right)$ possibilities.

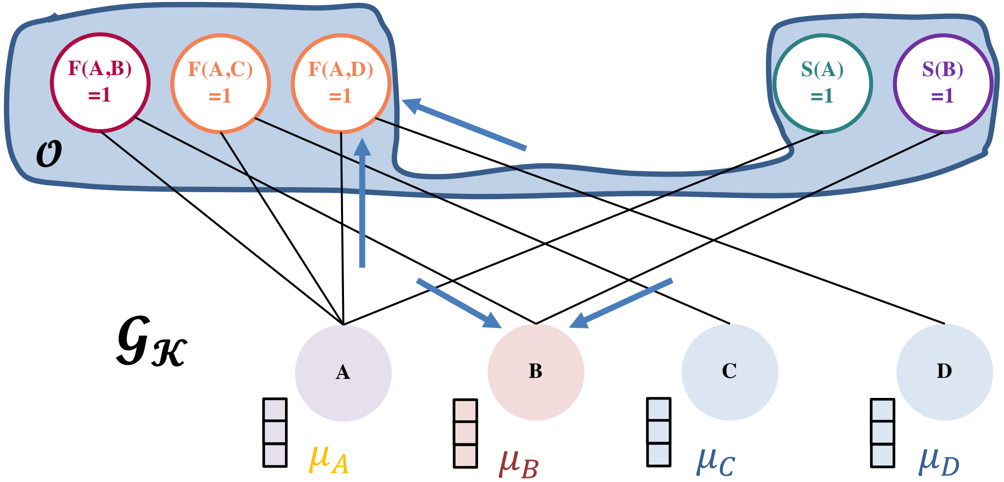


Efficient inference? Most previous works are on grounded networks

Use GNN for variational inference?

- GNN on original KB ($\mathcal{G}_{\mathcal{K}}$) to get embedding $\mu_A, \mu_B, \mu_C, \mu_D$ for entities

$$\mu_A, \mu_B, \mu_C, \mu_D = GNN(\mathcal{G}_{\mathcal{K}}; \theta)$$



Iterative update $t = 0, \ldots, T$:

$$\mu_B^{t+1} = MLP\left(\mu_B^t + \sum_{r \in N(B)} \mu_r^t\right)$$

$$\mu_r^{t+1} = MLP\left(\mu_r^t + \sum_{c \in N(r)} \mu_c^t\right)$$

Similar structure = similar GNN embedding (Dai ICML'16, Xu ICLR'19)

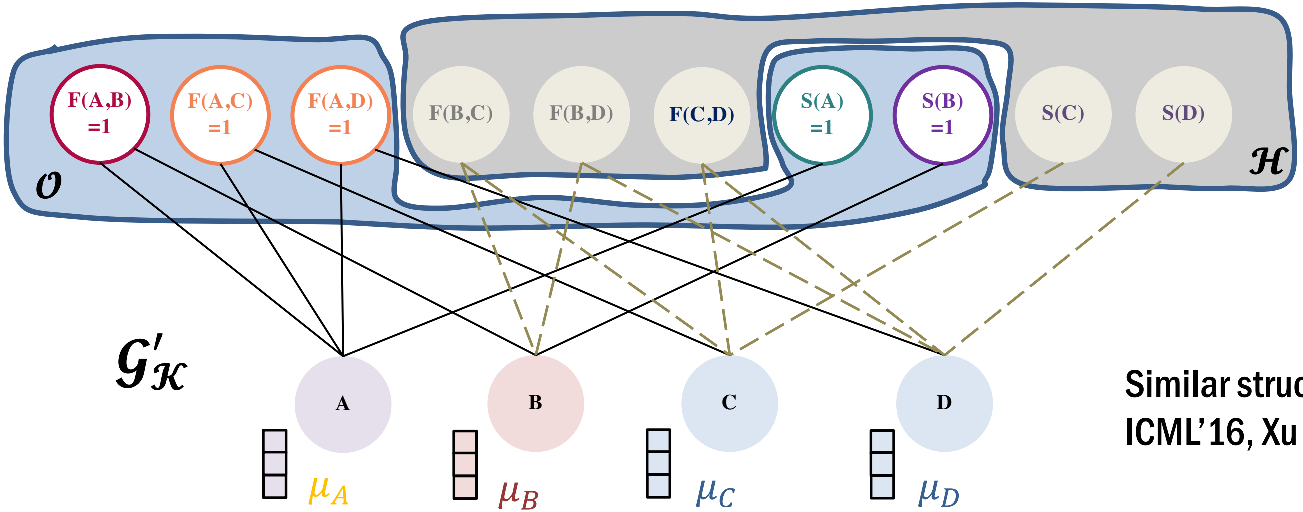
[Zhang et al. Arxiv 2019]

Use GNN for variational inference? (eg. Inside VAE)

- GNN on original KB ($\mathcal{G}_{\mathcal{K}}$) to get embedding $\mu_A, \mu_B, \mu_C, \mu_D$ for entities

$$\mu_A, \mu_B, \mu_C, \mu_D = GNN(\mathcal{G}_{\mathcal{K}}; \theta)$$

- Define $Q(\mathrm{F(B,C)} = 1|\mathcal{O}) = \frac{1}{1+\exp(\mu_B^{\top} \Theta_F\, \mu_C)}$, $Q(\mathrm{S(C)} = 1|\mathcal{O}) = \frac{1}{1+\exp(\theta_S^{\top}\, \mu_C)}$



Train the parameters

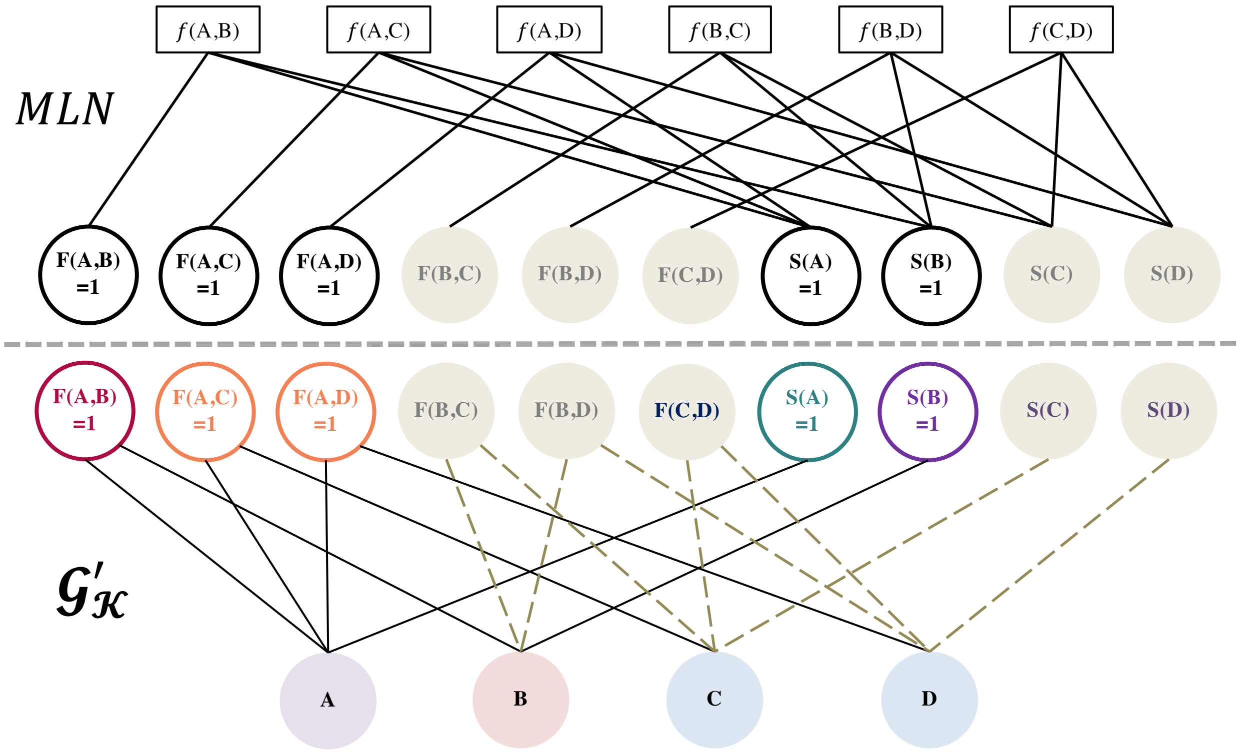
$$\theta, \Theta_F, \theta_S, ..$$

with mean field variational inference

Similar structure = similar GNN embedding (Dai ICML'16, Xu ICLR'19)

[Zhang et al. Arxiv 2019]

Is GNN embedding expressive enough?



$F(x,y) \overset{MLN}{\Longleftrightarrow} F(x',y')$

is this true?

$F(x,y) \overset{\mathcal{G}'_{\mathcal{K}}}{\Leftrightarrow} F(x',y')$

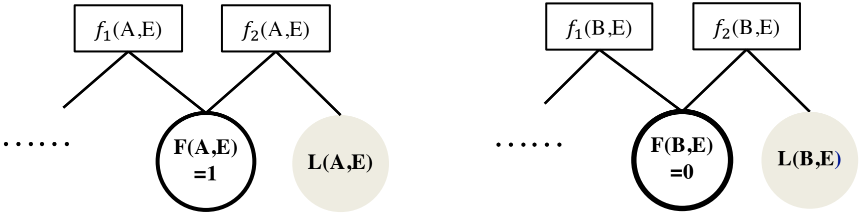
if and only if

$x \overset{\mathcal{G}_{\mathcal{K}}}{\Leftrightarrow} x'$ and $y \overset{\mathcal{G}_{\mathcal{K}}}{\Leftrightarrow} y'$

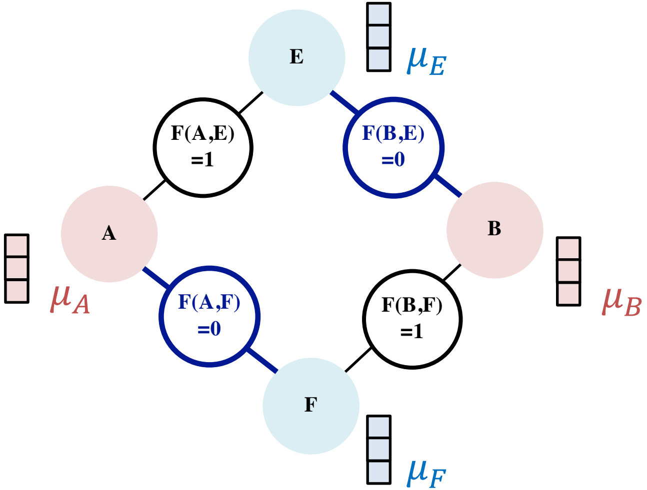
[Zhang et al. Arxiv 2019]

ExpressGNN: add a tunable embedding

- formula 1 f_1(A,B): Friend(A,B) ∧ Smoke(A) ⇒Smoke(B)
- formula 2 f_2(A,B): Friend(A,B) ⇒ Like(A,B)



L(A,E) and L(B,E) are different.



A and B are the same in KB.

$\mu_A, \mu_B, \mu_E, \mu_F = GNN(\mathcal{G}_{\mathcal{K}}; \theta)$

$\omega_A, \omega_B, \omega_E, \omega_F$ ←tunable low dimensional embedding

$$Q(\mathrm{L(A,E)} = 1|\mathcal{O}) = \frac{1}{1 + \exp(\mu_A^\top \Theta_L \mu_E + \omega_A^\top \omega_E)}$$

$$Q(\mathrm{L(B,E)} = 1|\mathcal{O}) = \frac{1}{1 + \exp(\mu_B^\top \Theta_L \mu_E + \omega_B^\top \omega_E)}$$

[Zhang et al. Arxiv 2019]

UW-CSE

- 22 relations
  - Teach, publish …

- Task goal
  - Predict who is whose advisor
  - Zero observed facts for query predicates

- 94 crowd-sourced FOL formulas

$\mathrm{advisedBy}(s, p) \Rightarrow \mathrm{professor}(p)$

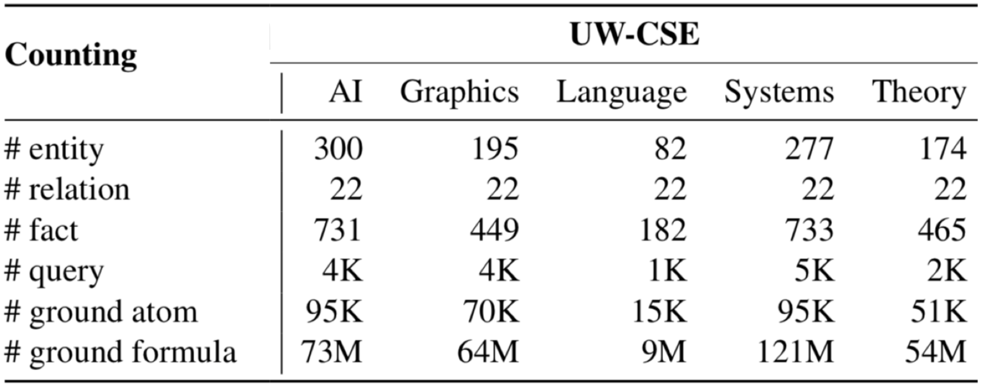
$\mathrm{advisedBy}(s, p) \Rightarrow \neg\mathrm{yearsInProgram}(s, \mathrm{Year\_1})$

$\mathrm{professor}(x) \Rightarrow \neg\mathrm{student}(y)$

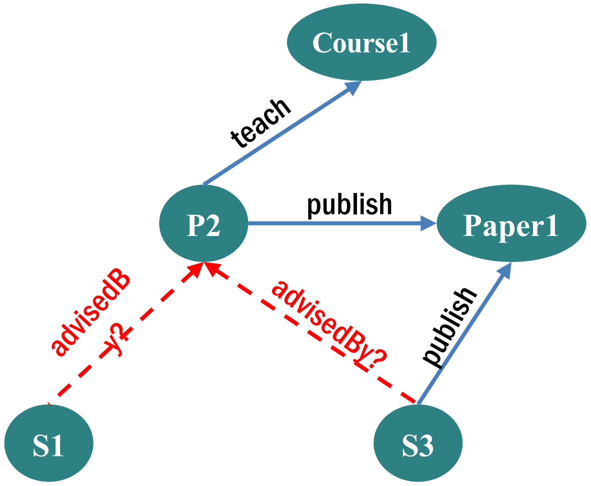
$\mathrm{publication}(p, x) \vee \mathrm{publication}(p,y) \vee \mathrm{student}(x) \vee \neg\mathrm{student}(y) \Rightarrow \mathrm{professor}(y)$

$\mathrm{student}(x) \vee \neg\mathrm{advisedBy}(x,y) \Rightarrow \mathrm{tempAdvisedBy}(x,y)$

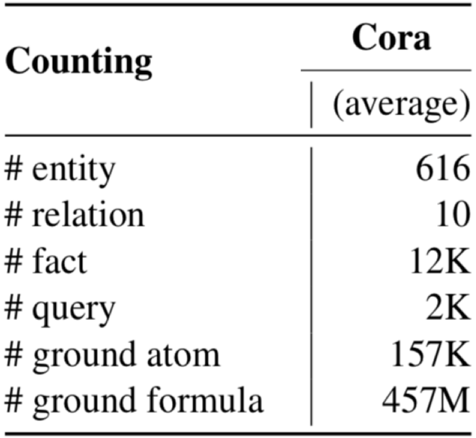
…



Counting	UW-CSE				
	AI	Graphics	Language	Systems	Theory
# entity	300	195	82	277	174
# relation	22	22	22	22	22
# fact	731	449	182	733	465
# query	4K	4K	1K	5K	2K
# ground atom	95K	70K	15K	95K	51K
# ground formula	73M	64M	9M	121M	54M



Cora dataset details (CS paper citatio



Counting	Cora
	(average)
# entity	616
# relation	10
# fact	12K
# query	2K
# ground atom	157K
# ground formula	457M

- 10 relation types
  - Author, Title, Venue, HasWordTitle, ...

- Task goal (entity resolution)
  - De-duplicate citations, authors, and venues
  - Zero observed facts for query predicates

- 46 crowd-sourced FOL formulas

Author(bc1,a1) v Author(bc2,a2) v SameAuthor(a1,a2) ⇒ SameBib(bc1,bc2)

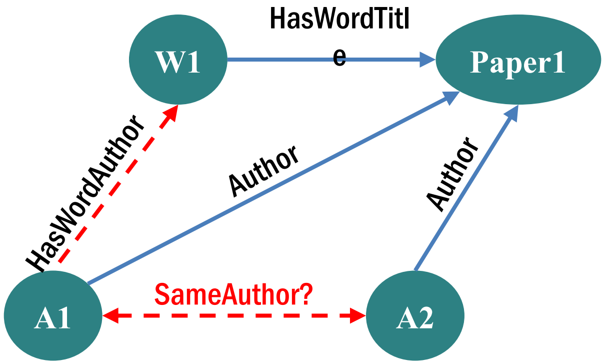
HasWordAuthor(a1, w) v HasWordAuthor(a2, w) ⇒ SameAuthor(a1, a2)

Title(bc1,t1) v Title(bc2,t2) v SameBib(bc1,bc2) ⇒ SameTitle(t1,t2)

SameVenue(v1,v2) v SameVenue(v2,v3) ⇒ SameVenue(v1,v3)

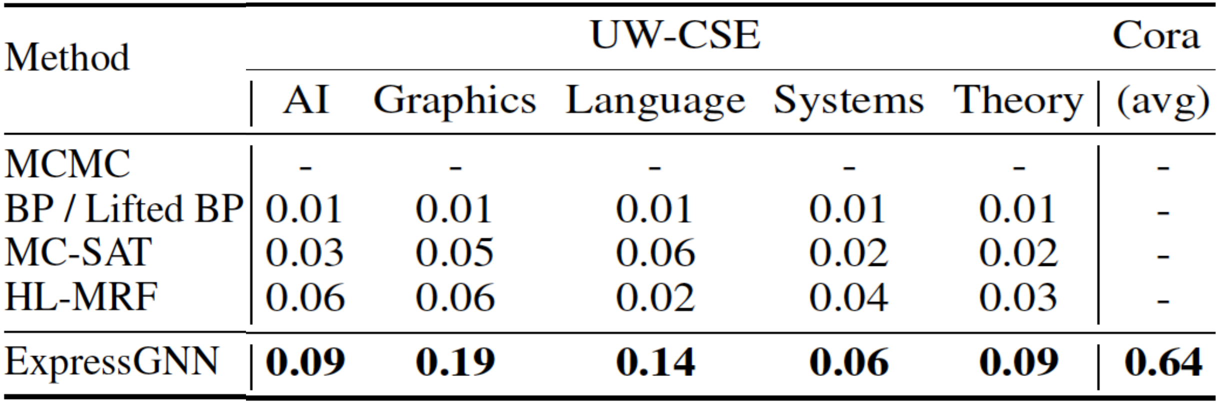
Title(bc1, t1) v Title(bc2, t2) v HasWordTitle(t1, +w) v HasWordTitle(t2, +w) ⇒ SameBib(bc1, bc2)

...



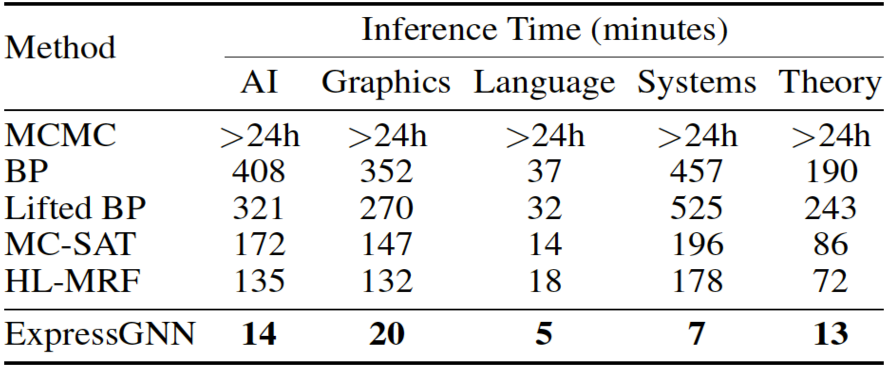
Inference accuracy and time

- Area under precision-recall curve (AUC-PR)



Method	UW-CSE					Cora
	AI	Graphics	Language	Systems	Theory	(avg)
MCMC	-	-	-	-	-	-
BP / Lifted BP	0.01	0.01	0.01	0.01	0.01	-
MC-SAT	0.03	0.05	0.06	0.02	0.02	-
HL-MRF	0.06	0.06	0.02	0.04	0.03	-
ExpressGNN	**0.09**	**0.19**	**0.14**	**0.06**	**0.09**	**0.64**

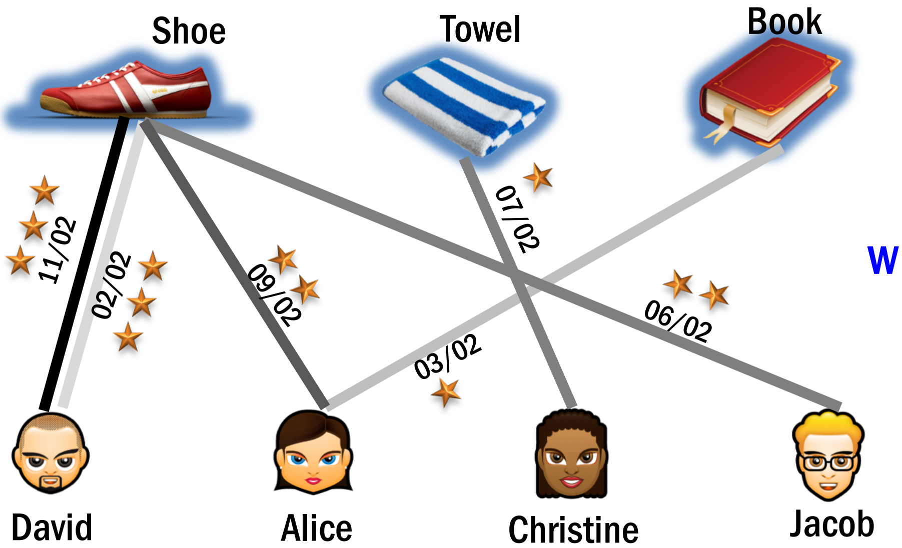
- Inference wall clock time



Method	Inference Time (minutes)				
	AI	Graphics	Language	Systems	Theory
MCMC	>24h	>24h	>24h	>24h	>24h
BP	408	352	37	457	190
Lifted BP	321	270	32	525	243
MC-SAT	172	147	14	196	86
HL-MRF	135	132	18	178	72
ExpressGNN	**14**	**20**	**5**	**7**	**13**

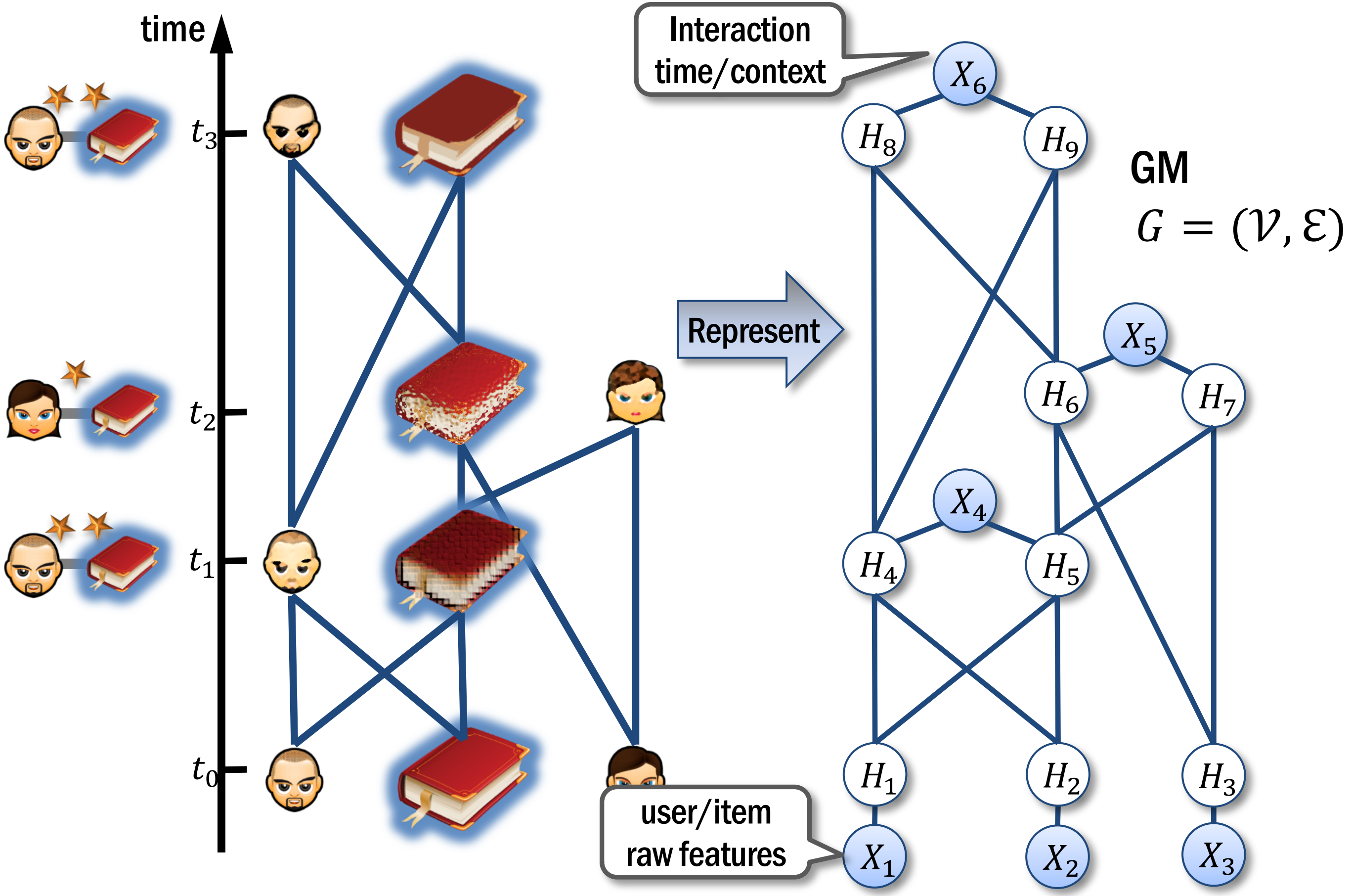
GNN to Parametrize Algorithm for Dynamic Networks

Dynamic processes over networks



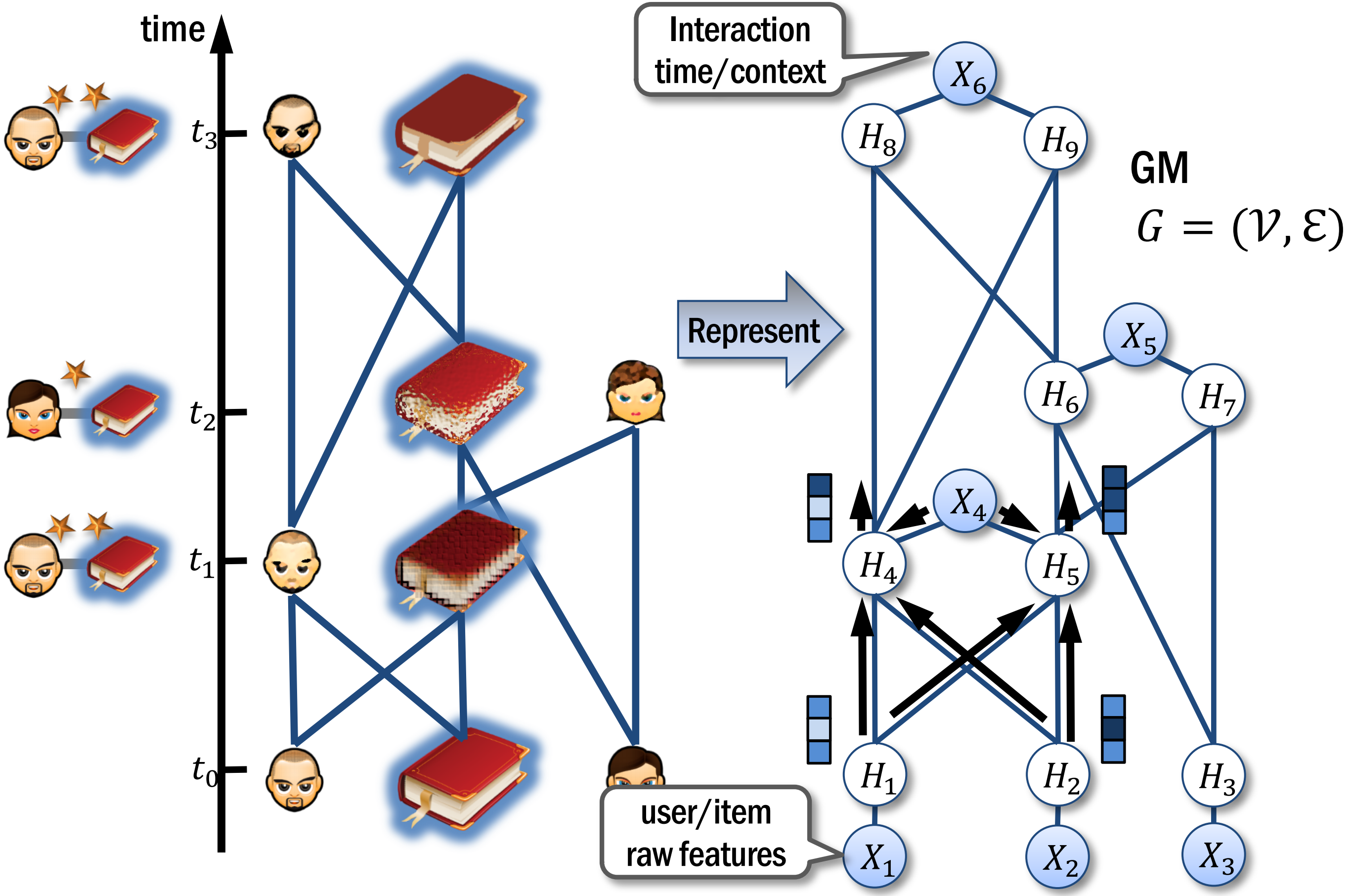
who will do **what** and **when**?

Unroll: time-varying dependency structure



[Dai, et al. Recsys 2016, Trivedi et al. ICML 2017]

Forward graph neural networks



Interaction time/context

Represent

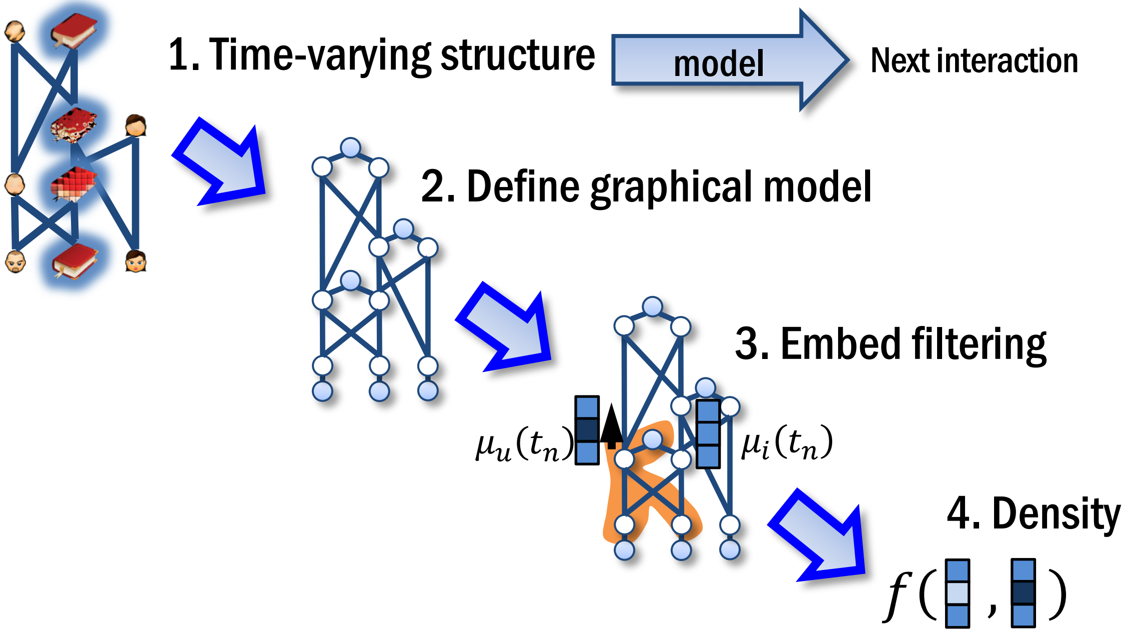
GM

$G = (\mathcal{V}, \mathcal{E})$

user/item raw features

[Dai, et al. Recsys 2016, Trivedi et al. ICML 2017]

Embedding filtering algorithm for generative model



1. Time-varying structure

2. Define graphical model

3. Embed filtering

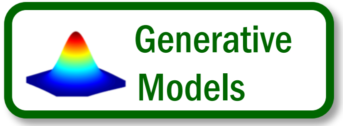
4. Density

Compatibility between user u and item i

$$\alpha_{ui} = \exp\left(\mu_u^\top(t_n)\mu_i(t_n)\right)$$

Likelihood of next event time $p_{ui}(t)$

$$\alpha_{ui}(t - t_n)\exp\left(-\frac{\alpha_{ui}(t - t_n)^2}{2}\right)$$



5. Train with MLE or GAN

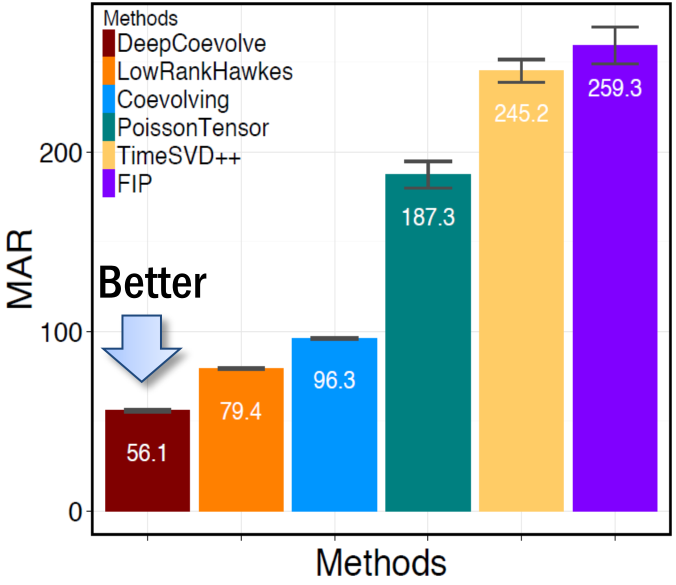
[Dai, et al. Recsys 2016, Trivedi et al. ICML 2017]

IPTV dataset

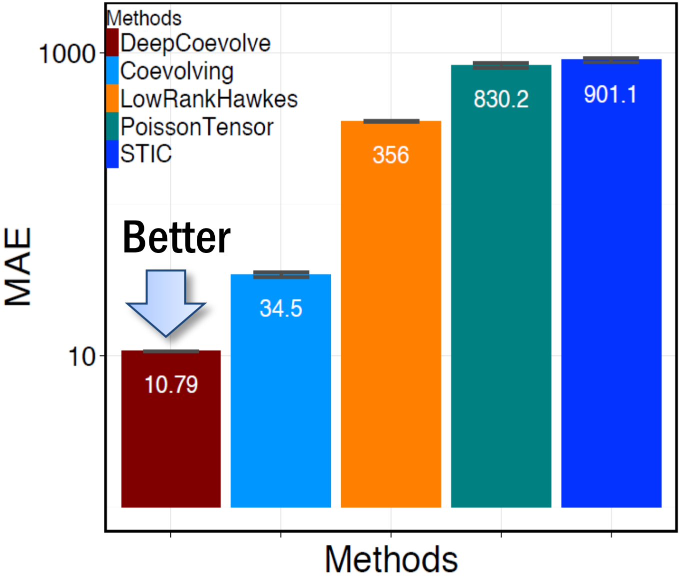
7,100 users, 436 programs, ~2M views

MAR: mean absolute rank difference

MAE: mean absolute error (hours)



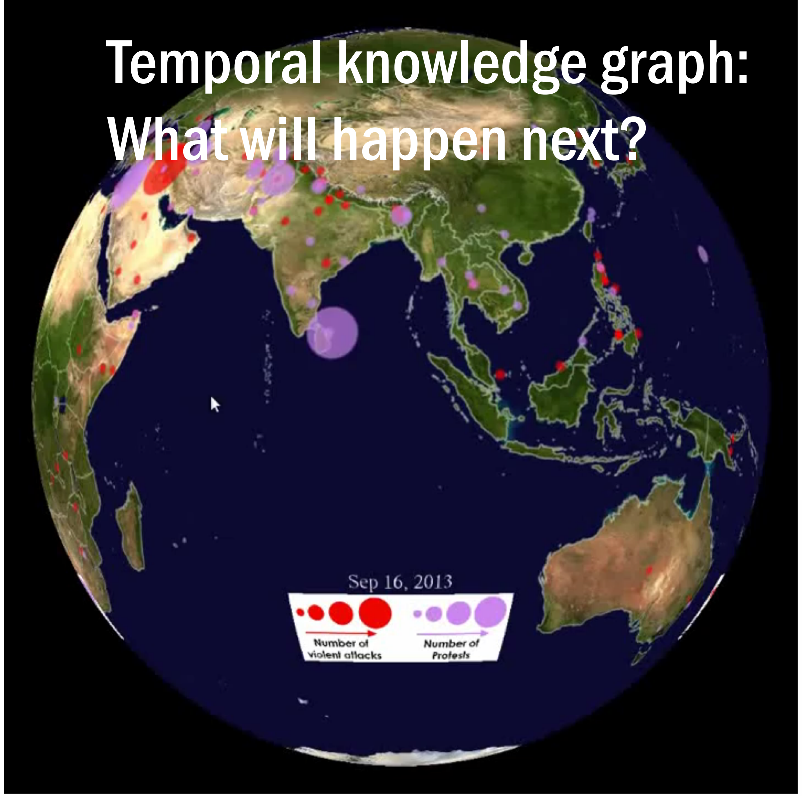
Next item prediction



Return time prediction

[Dai, et al. Recsys 2016]

Temporal knowledge graph: What will happen next?



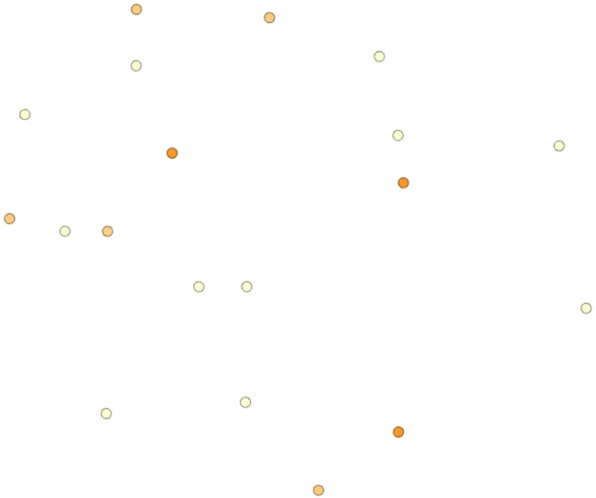
GDELT database

Events in news media

subject – relation – object

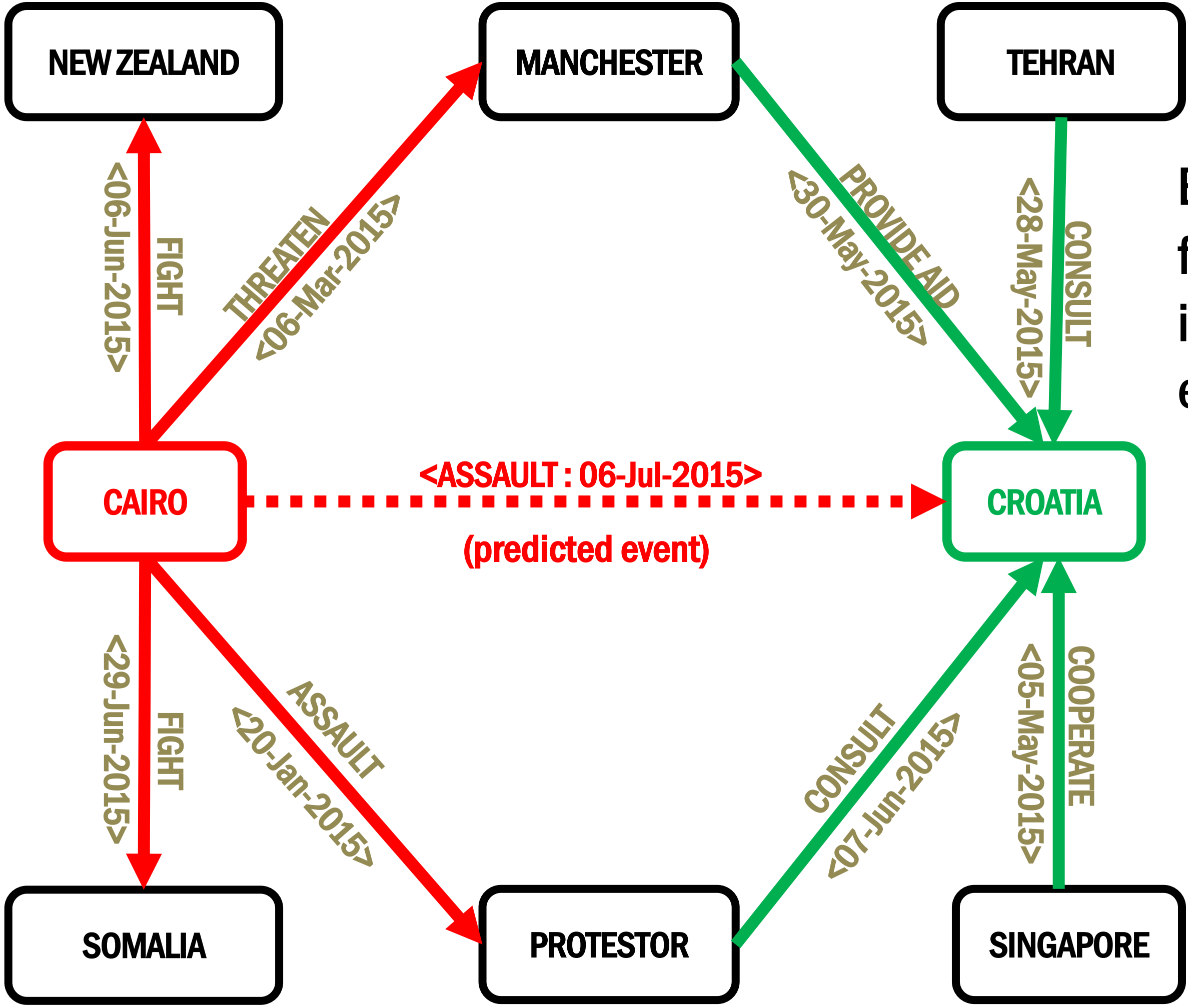
and time

Total archives span >215 years, trillion of events



Time-varying dependency structure

[Trivedi et al. ICML 2017]



NEW ZEALAND

MANCHESTER

TEHRAN

FIGHT <06-Jun-2015>

THREATEN <06-Mar-2015>

PROVIDE AID <30-May-2015>

CONSULT <28-May-2015>

CAIRO

<ASSAULT : 06-Jul-2015>

(predicted event)

CROATIA

FIGHT <29-Jun-2015>

ASSAULT <20-Jan-2015>

CONSULT <07-Jun-2015>

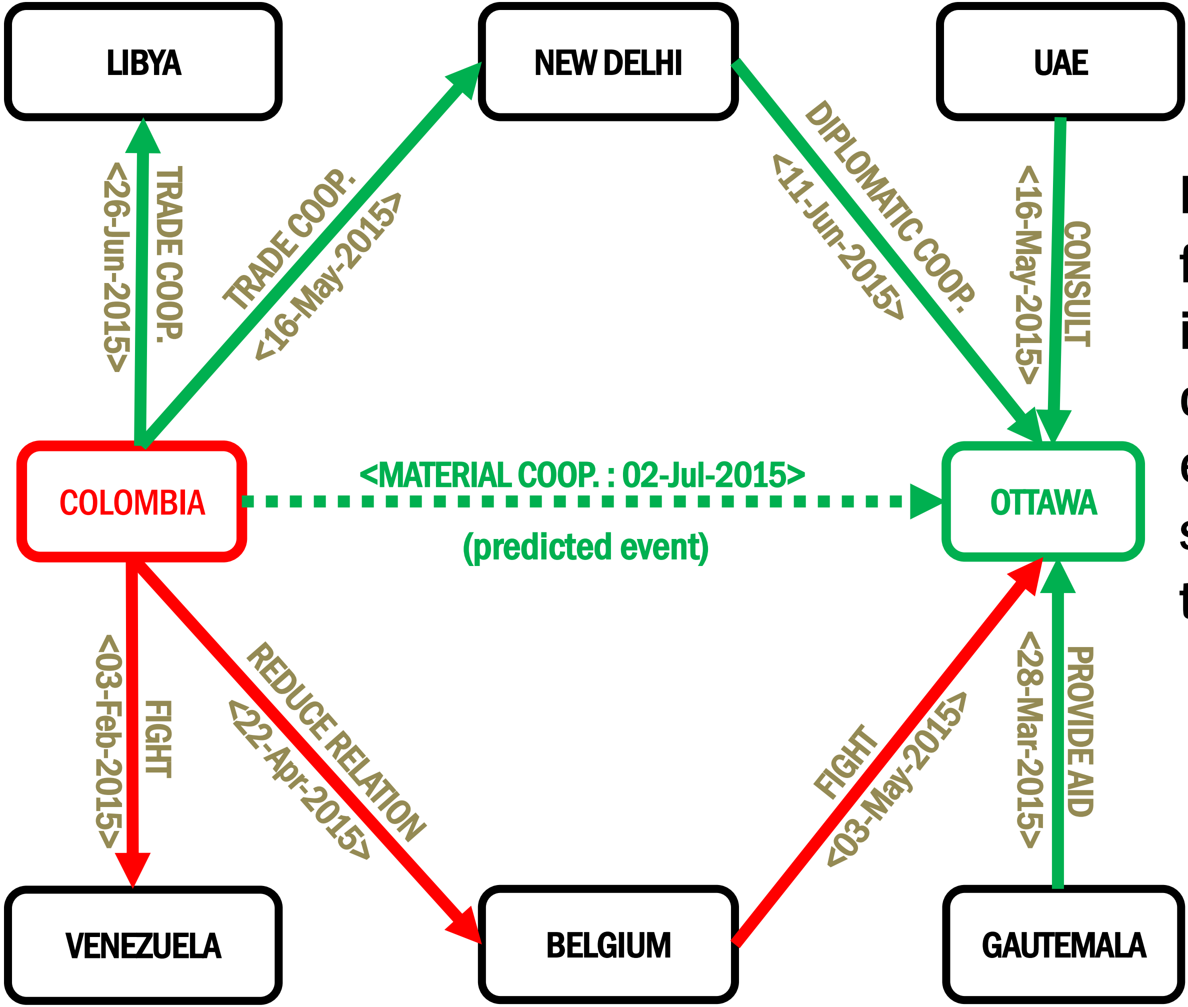
COOPERATE <05-May-2015>

SOMALIA

PROTESTOR

SINGAPORE

Enemy's friend is an enemy



LIBYA

NEW DELHI

UAE

TRADE COOP.

<26-Jun-2015>

TRADE COOP.

<16-May-2015>

DIPLOMATIC COOP.

<11-Jun-2015>

<16-May-2015>

CONSULT

COLOMBIA

<MATERIAL COOP. : 02-Jul-2015>

(predicted event)

OTTAWA

<03-Feb-2015>

FIGHT

REDUCE RELATION

<22-Apr-2015>

FIGHT

<03-May-2015>

<28-Mar-2015>

PROVIDE AID

VENEZUELA

BELGIUM

GAUTEMALA

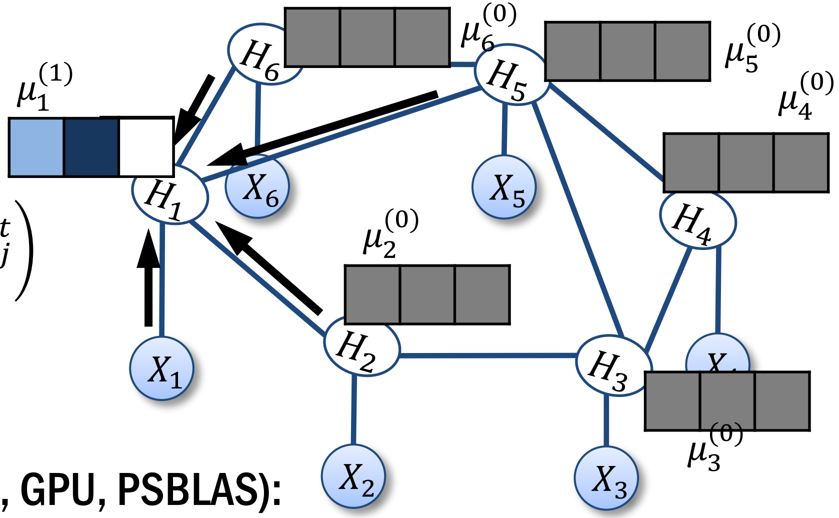
Friends'
friend
is a friend,
common
enemy
strengthen
the tie

Large Scale Implementation

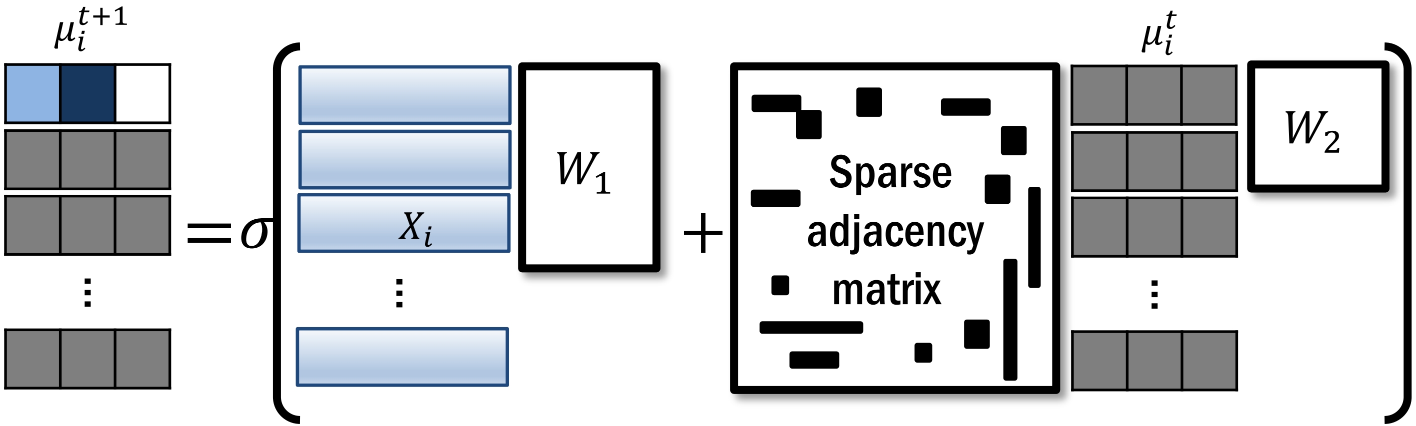
Sparse matrix formulation

Iterative node update formula:

$$\mu_i^{t+1} \leftarrow \sigma\left(\underset{\substack{\uparrow \\ d\times n \\ \text{matrix}}}{W_1} X_i + \underset{\substack{\uparrow \\ d\times d \\ \text{matrix}}}{W_2} \sum_{j\in\mathcal{N}(i)} \mu_j^t\right)$$



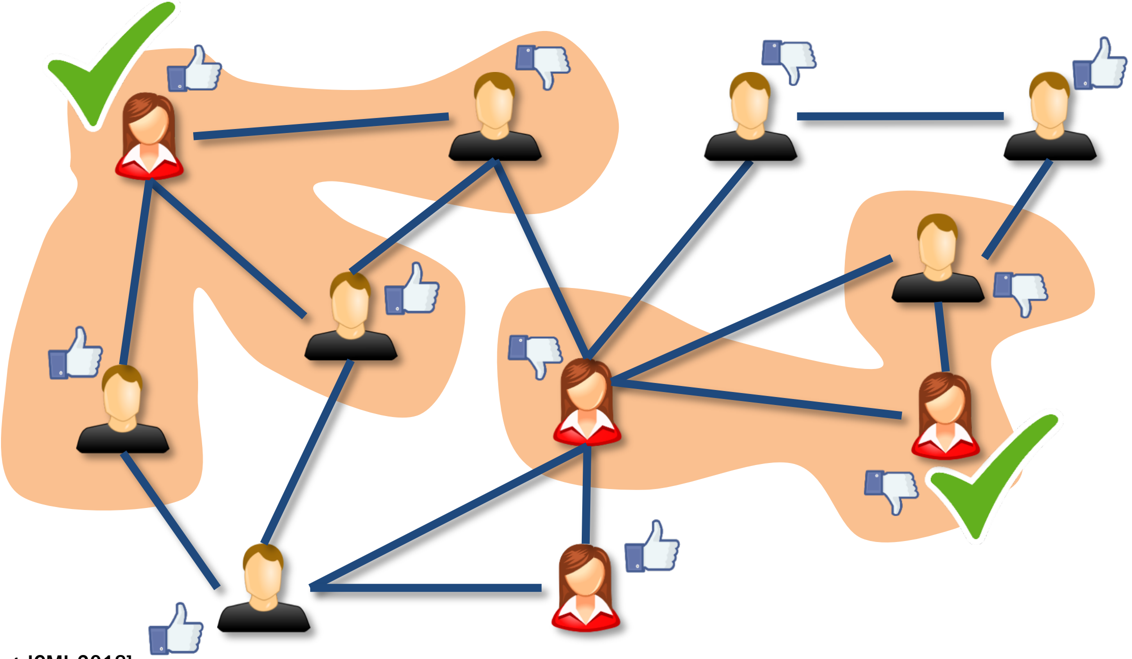
Sparse matrix operation (MKL, GPU, PSBLAS):



Stochastic training

1. Use a mini-batch of nodes for a stochastic loss function
2. Propagation step T determines the subgraph to compute
3. Embedding updates on subgraph

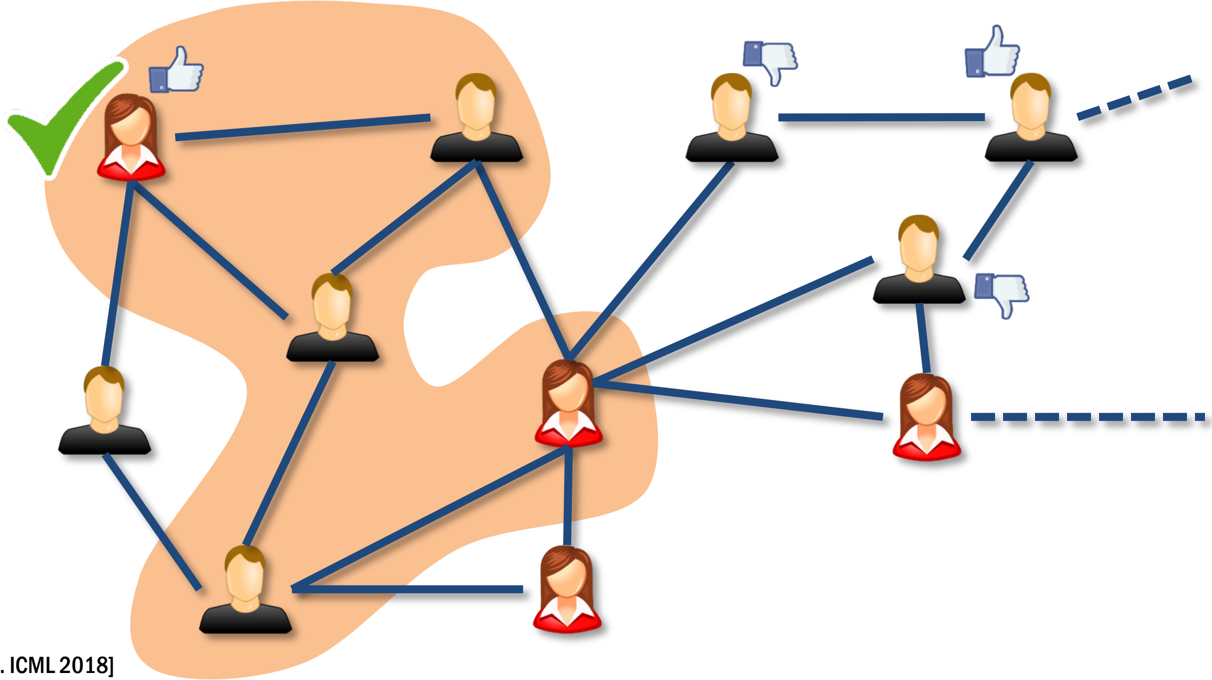
Eg. $T = 1$



[Chen, Zhu & Song. ICML 2018]

Doubly stochastic training

1. Sample a node
2. Propagation T steps to obtain a subgraph (eg. $T = 2$)
3. Subsample the subgraph
4. Compute loss using the subsampled subgraph



[Chen, Zhu & Song. ICML 2018]

Variance reduction training algorithm

- Intuition: model parameters W change slowly, so are embeddings

- Idea: approximate embeddings by their historical values

- Maintain history embedding $\bar{\mu}_j^{(t)}$, and $\Delta\mu_j^{(t)} = \mu_j^{(t)} - \bar{\mu}_j^{(t)}$

$$\sigma\left(W_1\mu_i^{(t)} + W_2 \underbrace{\sum_{j\in\mathcal{N}(i)} \mu_j^{(t)}}\right)$$

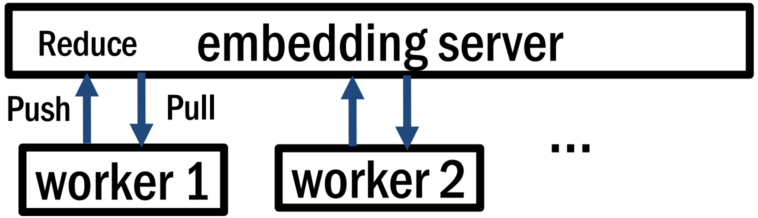
Rewrite $$= \sum_{j\in\mathcal{N}(i)} \left(\Delta\mu_j^{(t)} + \bar{\mu}_j^{(t)}\right)$$

Subsample $$\approx \frac{|\mathcal{N}(i)|}{|\mathcal{N}'(i)|} \sum_{j\in\mathcal{N}'(i)} \Delta\mu_j^{(t)} + \sum_{j\in\mathcal{N}(i)} \bar{\mu}_j^{(t)}$$

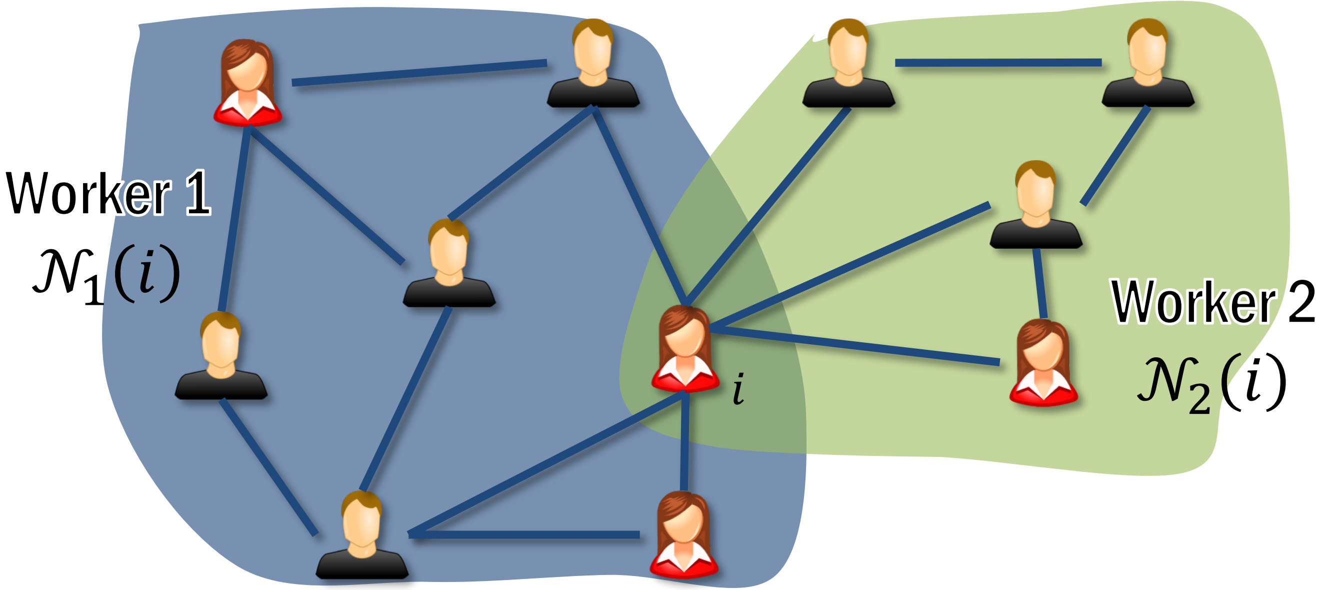
1. $\Delta\mu_j^{(t)} \to 0$
2. Variance $\to 0$
3. Gradient unbiased asymptotically

[Chen, Zhu & Song. ICML 2018]

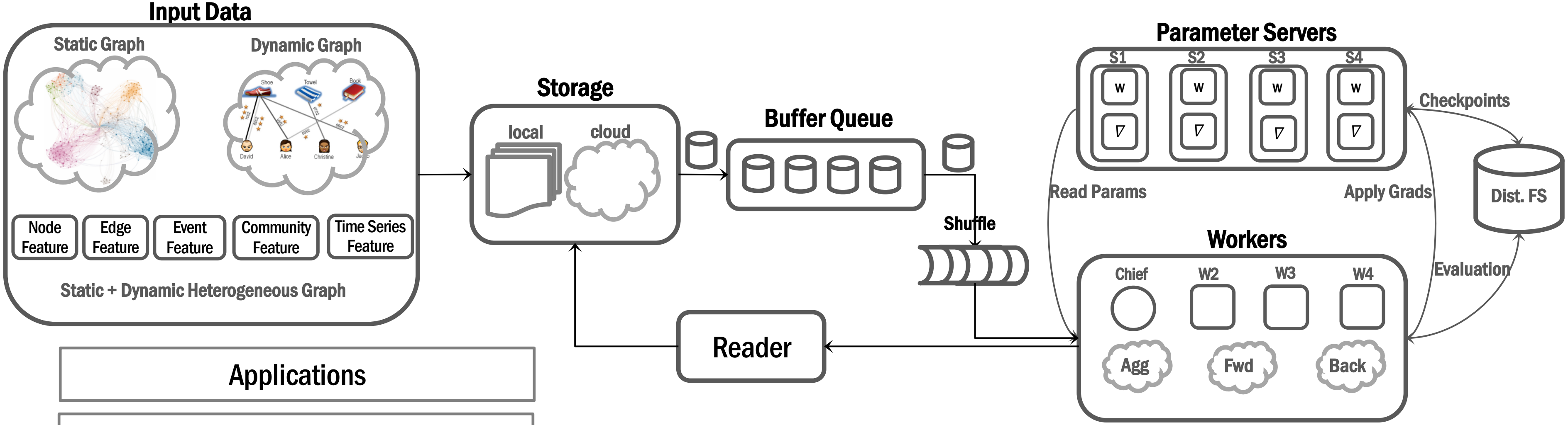
Parameter server architecture


Reduce
embedding server
Push
Pull
worker 1
worker 2
...

$$\mu_i^{t+1} \leftarrow \sigma\left(W_1 X_i + W_2 \sum_{j\in\mathcal{N}_1(i)} \mu_j^t + W_2 \sum_{j\in\mathcal{N}_2(i)} \mu_j^t\right)$$



Distributed Platform of Structure2Vec



Applications

Distributed Structure2Vec

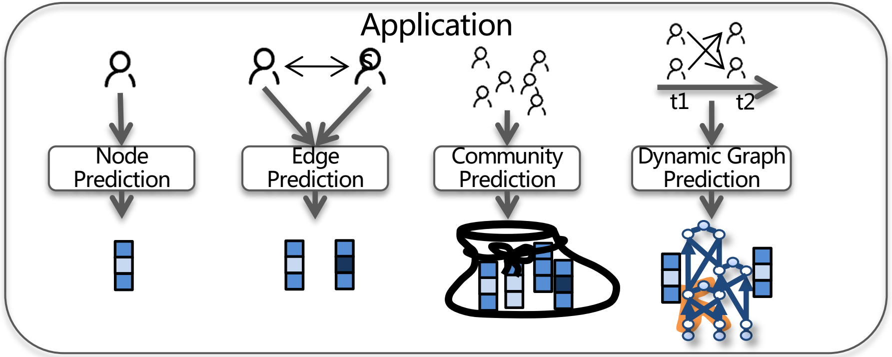
TensorFlow ODPS API Python

docker docker docker docker docker docker

Distributed Linux System

CPU CPU GPU GPU FPGA FPGA

Structure2Vec Architecture

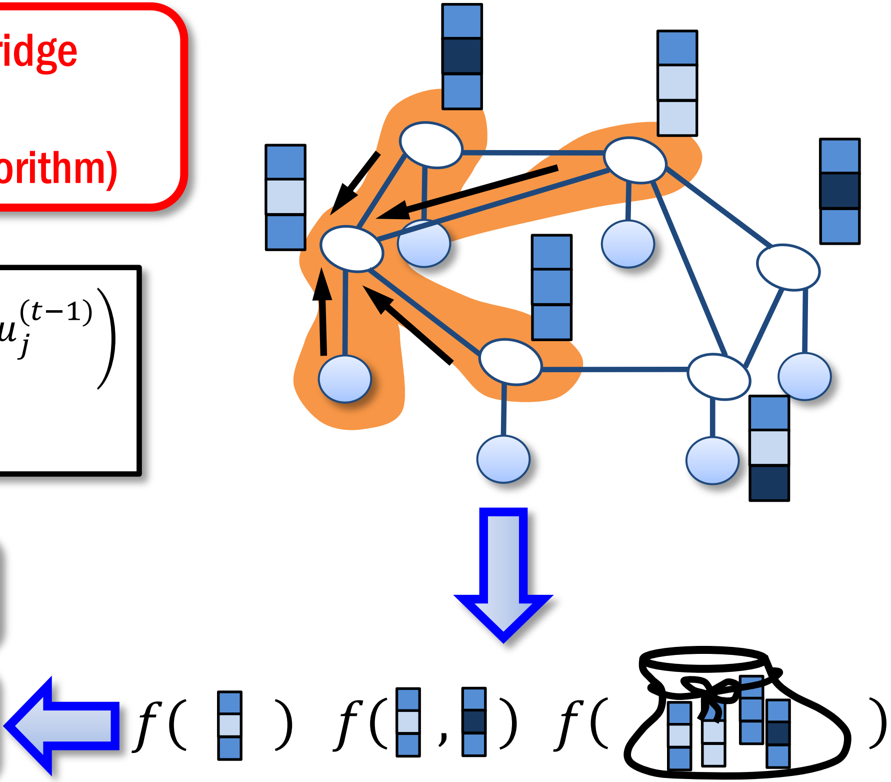


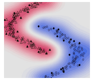
Conclusion

GCN/GNN/Structure2Vec = Parametrized algorithm

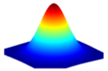
Open new possibility to bridge
Deep learning &
Structures (Graph, Logic, Algorithm)

$$\mu_i^{(t)} \leftarrow \sigma\left(W_1 X_i + W_2 \sum_{j \in \mathcal{N}(i)} \mu_j^{(t-1)}\right)$$





Supervised Learning



Generative Models

Reinforcement Learning

References

- H. Dai, B. Dai and L. Song. Structure2Vec: Discriminative Embedding of Latent Variable Models for Structured Data, ICML 2016.
- Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi and L. Song. Can Graph Neural Networks Help Logic Reasoning? Arxiv 2019.
- J. Chen, J. Zhu, and L. Song. Stochastic Training of Graph Convolutional Networks. ICML 2018.
- R Trivedi, H Dai, Y Wang, L Song. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graph. ICML 2017.
- H Dai, Y Wang, R Trivedi, L Song. Deep Coevolutionary Network: Embedding User and Item Features for Recommendation. Recsys Workshop on Deep Learning. 2017. (BEST PAPER)
- H. Dai, E. Khalil, Y. Zhang, B. Dilkina and L. Song. Learning Combinatorial Optimizations over Graphs. NIPS 2017.
- X. Xu, C. Liu, Q. Feng, H. Yin, L. Song, D. Song. Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection. CCS. 2017.
- H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song. Learning Steady States of Iterative Algorithms over Graphs. ICML 2018.
- H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial Attack on Graph Structured Data. ICML 2018.
- Y. Zhang, H. Dai, Z. Kozareva, A. Smola, and L. Song. Variational Reasoning for Question Answering with Knowledge Graph. AAAI 2018.
- Z. Liu, C. Chen, X. Yang, J. Zhou, X. Long and L. Song . Heterogeneous Graph Neural Networks for Malicious Account Detection. CCS2017, CIKM 2018.
- Z. Liu, C. Chen, L. Li, J. Zhou, X. Li and L. Song. Geniepath: Graph Neural Networks with Adaptive Receptive Paths. AAAI 2019.
- H. Dai, Y. Tian, B. Dai, S. Skiena and L. Song. Syntax Directed Variational Autoencoder for Structured Data. ICLR 2018.
- X. Si, H. Dai, M. Raghothanman, M. Naik and L. Song. Learning Loop Invariants for Program Verification. NIPS 2018.