图神经网络导论

富信息图神经网络

授课教师：周晟

浙江大学　软件学院

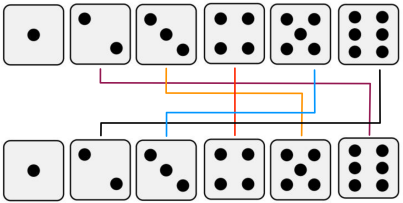
2025.12.2

课程内容

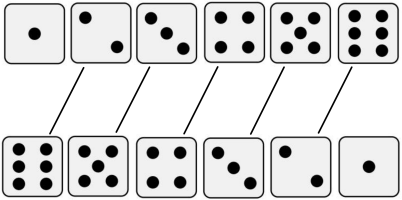
1. 上节课回顾
2. 研究背景
3. 有向图神经网络
4. 异构图神经网络
5. 动态图神经网络

生活中的卷积

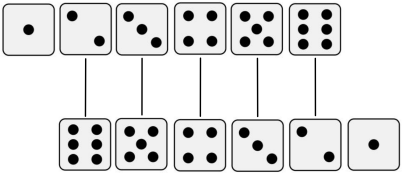
生活中的卷积



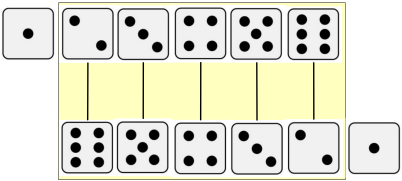
解



卷



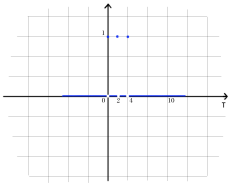
移



积

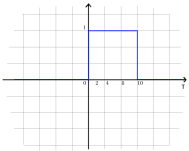
生活中的卷积

肥宅快乐问题



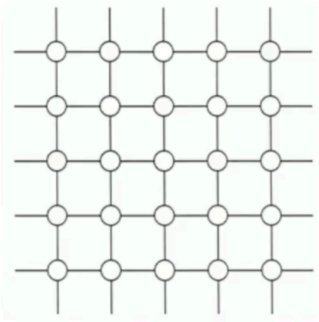
间隔的快乐

肥宅快乐问题



连续的快乐

从图像卷积到图卷积



Grid-like network

Irregular network

图卷积的困难

1. 节点没有顺序（排列不变性）

图卷积

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

两个函数的卷积可以写成傅里叶变换的形式。

图上的卷积

1. 定义图上的傅立叶变换/逆变换
2. 定义图上的信号 f
3. 定义图上的操作 g（学习目标）

图上的卷积

- 卷积可以写成傅里叶变换的形式

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

- 图傅里叶变换及其逆变换：

$$\mathcal{GF}\{x\} = U^T x \quad , \quad \mathcal{IGF}\{x\} = Ux$$

- 图上的卷积可以表示为：

$$x * y = U\left(\left(U^T x\right) \odot \left(U^T y\right)\right)$$

$$x * y = U g_\theta U^T x$$

课程内容

富信息网络

图的数学定义

图一般定义为：$G = \{V, E, X\}$，其中 V 是节点的集合，E 是边的集合，X 是节点属性的集合。

富信息网络

富信息网络是指网络结构、节点属性、边属性等信息丰富的网络。大多数真实世界的网络都是富信息网络

大部分真实世界的网络均为富信息网络，因此富信息图神经网络近年来得到了广泛的研究。

富信息网络

计算机网络

社交网络

交通网络

神经元网络

卫星网络

富信息图神经网络

困难与挑战

1. 富信息网络中的节点关系更为复杂，邻近性和相似度的度量更为多样
2. 不同类型富信息网络需要设计不同的图神经网络结构

常见的富信息图神经网络

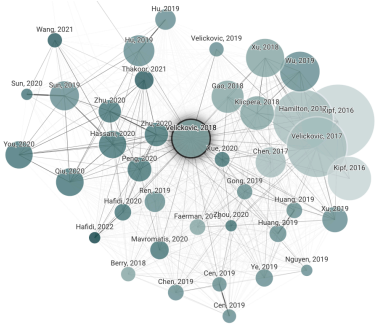
1. 有向图
2. 异构图
3. 动态图
4. ...

常见的有向网络

有向网络/图

有向网络（Directed Network/Graph）是指边带有方向的网络。有向网络中的边除了包含临近信息外，往往还包含层次或语义信息。

社交网络



引用网络

有向图神经网络的挑战

有向图的特点

1. 节点间临近关系非对称 (Asymmetric Proximity)
2. 网络中存在层次性等特殊结构

有向图神经网络的难点

1. 邻接矩阵非对称
2. 邻居关系多样化
3. 有向特征难以捕获

有向图神经网络的解决思路

如何定义有向网络特有的信息（What）？

1. Asymmetric Proximity
2. (Local) Hierarchical Structure
3. ...

对于有向网络特有的信息如何建模（How）？

1. Source/Target Embedding
2. 矩阵分解
3. Auto-Encoder
4. Graph Neural Networks
5. ...

有向图表征学习分类

1. Matrix Factorization Based
    1. HOPE[Ou et al., 2016]
2. Random Walk Based
    1. APP[Zhou et al., 2017]
    2. NERD[Khosla et al., 2019]
    3. InfoWalk[Zhou et al., 2021]
3. Auto-Encoder Based
    1. Gravity[Salha et al., 2019]
4. Graph Neural Networks Based

Asymmetric Transitivity Preserving Graph Embedding(HOPE)[1]

研究动机

图上的传递性（Transitivity）不一定是对称的（Asymmetric），这种非对称的传递性需要被保留到节点表征中去。

Table 1: General Formulation for High-order Proximity Measurements

Proximity Measurement	$\mathbf{M}_g$	$\mathbf{M}_l$
Katz	$\mathbf{I} - \beta \cdot \mathbf{A}$	$\beta \cdot \mathbf{A}$
Personalized Pagerank	$\mathbf{I} - \alpha\mathbf{P}$	$(1-\alpha) \cdot \mathbf{I}$
Common neighbors	$\mathbf{I}$	$\mathbf{A}^2$
Adamic-Adar	$\mathbf{I}$	$\mathbf{A} \cdot \mathbf{D} \cdot \mathbf{A}$

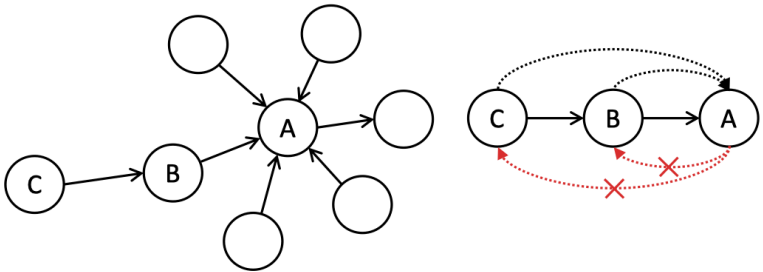
使用矩阵分解方法，使得每个节点有两套表征。

[1]Asymmetric Transitivity Preserving Graph Embedding(KDD 2016)

Scalable Graph Embedding for Asymmetric Proximity (APP)

研究动机

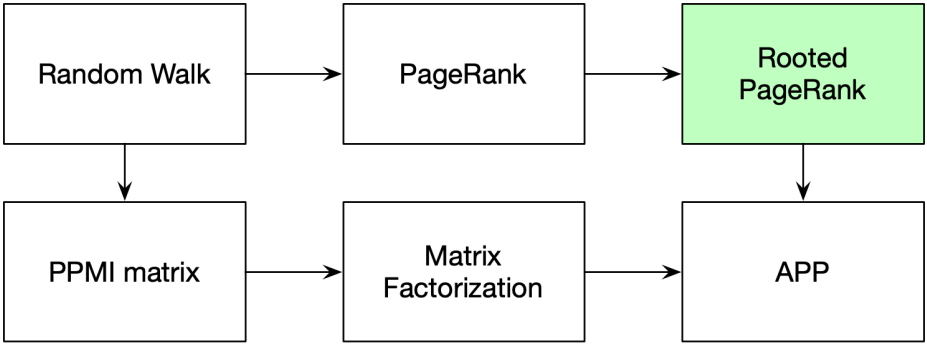
图表征学习的核心是节点之间的临近性（Proximity），然而有向图中的临近性是非对称的（Asymmetric）



Asymmetric Proximity[2]

[2]Scalable Graph Embedding for Asymmetric Proximity(AAAI 2017)

APP



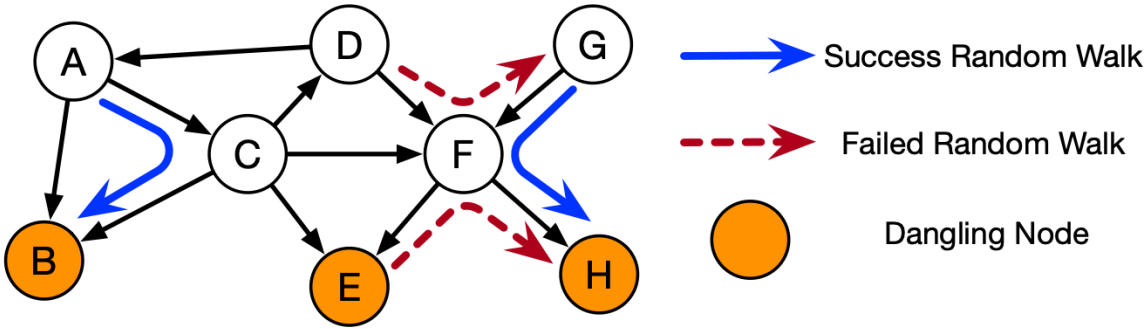
$$\text{Sample} \left\{ \sum_u \sum_v \left(\text{Sim}_u(v) \cdot \log \sigma \left(\overrightarrow{s_u} \cdot \overrightarrow{t_v} \right) + \frac{k}{|V|} \cdot \log \sigma \left(-\overrightarrow{s_u} \cdot \overrightarrow{t_v} \right) \right) \right\}$$

DataSet	arxiv						amazon					
	P@10	R@10	P@20	R@20	P@50	R@50	P@10	R@10	P@20	R@20	P@50	R@50
CNbrs	0.092	0.493	0.065	0.635	0.044	0.793	0.080	0.551	0.052	0.677	0.034	0.773
Adar	0.110	0.587	**0.082**	0.712	**0.051**	0.819	0.084	0.579	0.056	0.686	0.036	0.753
Jaccard	0.053	0.259	0.042	0.319	0.033	0.364	0.085	0.579	0.056	0.686	**0.037**	0.735
DeepWalk	0.095	0.598	0.060	0.757	0.029	0.894	0.061	0.423	0.038	0.534	0.018	0.633
Line	0.080	0.551	0.054	0.659	0.024	0.772	0.021	0.100	0.012	0.172	0.011	0.350
Node2Vec	0.082	0.508	0.052	0.660	0.026	0.823	0.071	0.497	0.047	0.659	0.023	0.809
APP	**0.122**	**0.697**	0.065	**0.829**	0.035	**0.973**	**0.095**	**0.660**	**0.059**	**0.824**	0.027	**0.928**

Table 5: Precision and Recall for top-k Node Recommendation, Undirected Graph

DataSet	epinions						cora					
	P@10	R@10	P@20	R@20	P@50	R@50	P@10	R@10	P@20	R@20	P@50	R@50
CNbrs	0.023	0.061	0.017	0.093	0.011	0.148	0.023	0.142	0.017	0.198	0.011	0.268
Adar	**0.026**	0.072	**0.018**	0.103	0.012	0.161	0.023	0.142	0.017	0.198	0.011	0.268
Jaccard	0.021	0.057	0.016	0.086	0.011	0.137	0.021	0.134	0.016	0.185	0.011	0.259
DeepWalk	0.015	0.049	0.011	0.066	0.006	0.094	0.025	0.180	0.019	0.270	0.011	0.400
Line	0.004	0.014	0.004	0.025	0.003	0.049	0.012	0.084	0.009	0.124	0.005	0.184
Node2Vec	0.010	0.032	0.007	0.045	0.004	0.067	0.022	0.160	0.018	0.256	0.010	0.400
APP	0.024	**0.075**	**0.018**	**0.112**	**0.013**	**0.182**	**0.030**	**0.223**	**0.023**	**0.340**	**0.014**	**0.525**

Direction-Aware User Recommendation Based on Asymmetric Network Embedding[3]

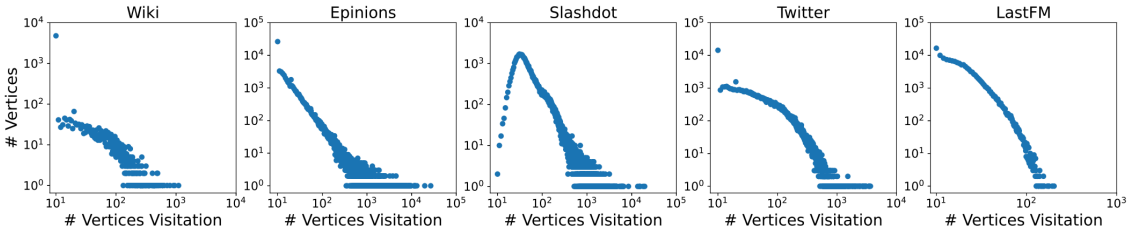


有向网络中的随机游走

[3]Direction-Aware User Recommendation Based on Asymmetric Network Embedding(TOIS 2021)

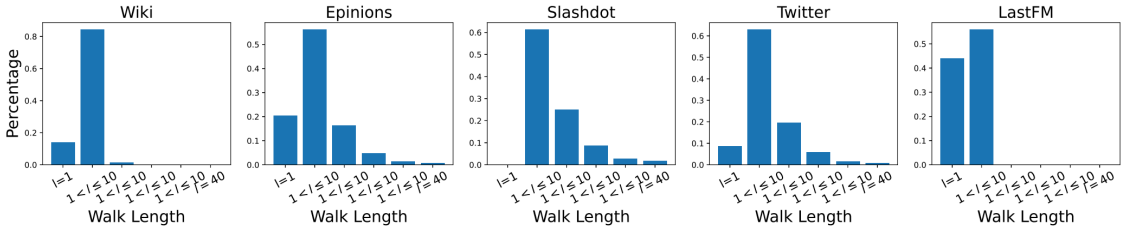
Direction-Aware User Recommendation Based on Asymmetric Network Embedding

在有向网络中使用传统随机游走的问题：



节点被访问次数问题

在有向网络中使用传统随机游走的问题：



随机游走长度问题

InfoWalk 随机游走

- 每一步随机游走时忽略边的方向

$$P\left(\mathcal{R}_{v_i}^{k+1}=b \mid \mathcal{R}_{v_i}^{k}=a\right)=\begin{cases}\frac{1}{d_a^{\text{out}}+d_a^{\text{in}}} & \mathrm{E}_{ab}=1 \text{ or } \mathrm{E}_{ba}=1\\ 0 & \text{otherwise}\end{cases}$$

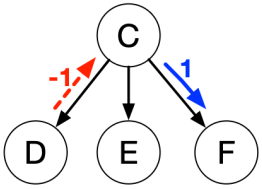
- 每一步随机游走时边的方向使用参数记录信息

$$r_{i,i+1}=\begin{cases}1 & \text{if } \mathrm{E}_{i,i+1}=1 \text{ and } \mathrm{E}_{i+1,i}=0\\ -1 & \text{if } \mathrm{E}_{i,i+1}=0 \text{ and } \mathrm{E}_{i+1,i}=1\\ 0 & \text{if } \mathrm{E}_{i,i+1}=1 \text{ and } \mathrm{E}_{i+1,i}=1\end{cases}$$

- 随机游走得到带参序列 $\mathcal{R}_{v_i}: v_i \xrightarrow{r_{i,j}} v_j \xrightarrow{r_{j,j+1}} \dots \xrightarrow{r_{k-1,k}} v_k$

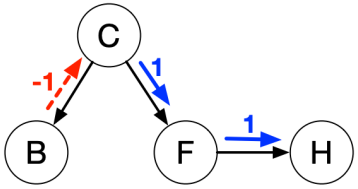
$$s_{i,i+k}=\frac{1}{k}\sum_{j=i}^{i+k-1} r_{j,j+1}$$

InfoWalk 随机游走



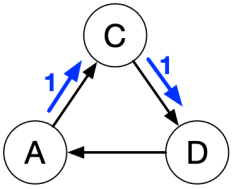
$S_{D\to C\to F} = 0$

(a)



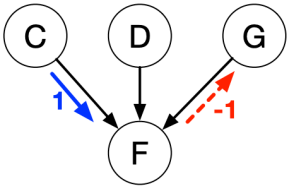
$S_{B\to C\to F\to H} = 1$

(c)

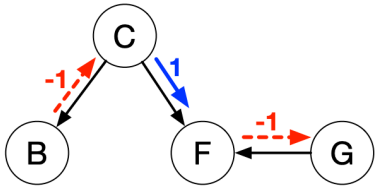


$S_{A\to C\to D} = 2$

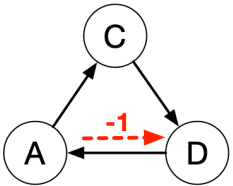
(e)



$S_{C\to F\to G} = 0$

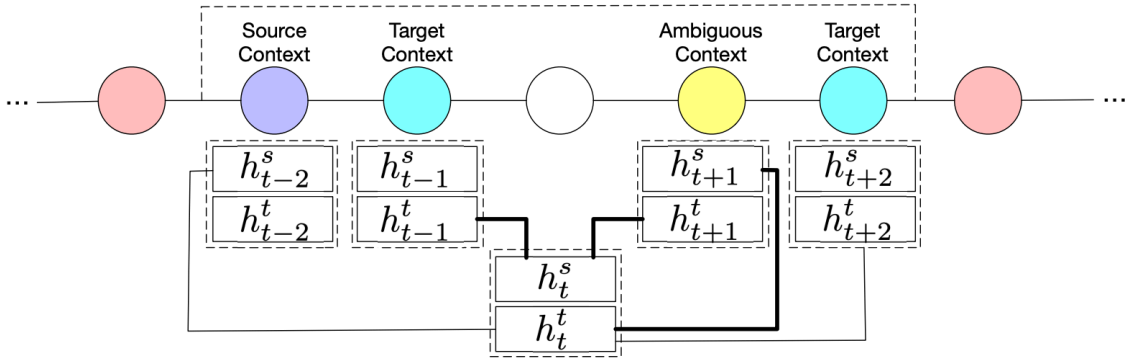


$S_{B\to C\to F\to G} = -1$



$S_{A\to D} = -1$

InfoWalk 随机游走



基于 InfoWalk 的节点分类

$$\max_{\mathrm{H}^s,\mathrm{H}^t} \sum_{u \in \mathrm{V}} \sum_{v \in DC_u} \log P\left(v \mid u, s_{u,v}\right)$$

基于 InfoWalk 的节点表征学习

- 定性学习

$$P\left(v \mid u, s_{u,v} > 0\right) = \frac{\exp\left(h_u^s \cdot h_v^t\right)}{\sum_{k \in V} \exp(h_u^s \cdot h_v^t)}$$
$$P\left(v \mid u, s_{u,v} < 0\right) = \frac{\exp\left(h_v^s \cdot h_u^t\right)}{\sum_{k \in V} \exp(h_v^s \cdot h_u^t)}$$
$$P\left(v \mid u, s_{u,v} = 0\right) = \frac{\exp\left(h_v^s \cdot h_u^t + h_u^s \cdot h_v^t\right)}{\sum_{k \in V} \exp(h_v^s \cdot h_u^t + h_u^s \cdot h_v^t)}$$

- 定量学习

$$\pi_{u,v} = \log\left(\frac{s_{u,v}+1}{v-u} + b\right)$$

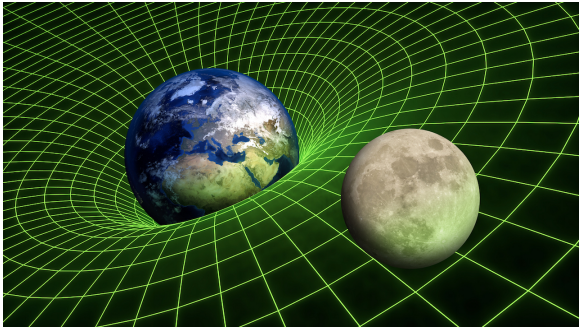
$$\max_{\mathrm{H}^s, \mathrm{H}^t} \sum_{u \in \mathrm{V}} \sum_{v \in DC_u} \log P\left(v \mid u, \pi_{u,v}\right) = \log \frac{\pi_{u,v} \cdot \exp\left(h_u^s \cdot h_v^t\right)}{\sum_{k \in V} \exp\left(h_u^s \cdot h_v^t\right)}$$

基于 InfoWalk 的节点表征学习

Dataset	Wiki		Epinions		Slashdot		Twitter		LastFM	
Metric	AUC	MAP	AUC	MAP	AUC	MAP	AUC	MAP	AUC	MAP
Node2Vec	0.855	0.805	0.853	0.84	0.738	0.740	0.874	0.910	0.923	0.933
DeepWalk	0.69	0.638	0.585	0.584	0.390	0.4155	0.852	0.892	0.825	0.838
GraRep	0.905	0.893	NA	NA	NA	NA	NA	NA	NA	NA
LINE	0.913	0.917	0.857	0.894	0.764	0.7909	0.791	0.8375	0.898	0.923
HOPE	0.93	0.948	0.889	0.924	0.777	0.8524	0.801	0.8417	NA	NA
APP	0.919	0.907	0.898	0.928	0.868	0.8877	0.873	0.918	0.926	0.935
ATP	0.85	0.779	NA	NA	NA	NA	NA	NA	NA	NA
Gravity	0.955	0.927	NA	NA	NA	NA	NA	NA	NA	NA
NERD	0.517	0.565	0.818	0.872	0.832	0.8767	0.694	0.742	0.744	0.773
GraphSAGE	0.938	0.917	**0.930**	**0.942**	**0.886**	0.895	0.849	0.8875	0.948	0.950
GAT	0.839	0.785	0.786	0.776	0.631	0.569	0.821	0.862	0.909	0.914
GREED	0.793	0.725	0.633	0.543	0.720	0.712	0.650	0.666	0.826	0.828
ShortWalk	0.708	0.673	0.787	0.805	0.638	0.660	0.889	0.920	0.899	0.913
DNE-L	0.960	0.955	0.926	0.939	0.863	**0.899**	**0.899**	**0.928**	**0.951**	**0.956**
DNE-T	**0.968**	**0.963**	0.929	0.941	0.857	0.896	0.889	0.921	0.946	0.951
Impv%	†0.8%	†0.8%	—	—	—	†0.4%	†2.9%	†1.0%	†0.3%	†0.6%

Negative links contains the reverse direction of positive edges. NA denotes the methods that cannot run on our hardware setup. † indicates that the result of a paired difference test is significant at $p < 0.05$.

Gravity-Inspired Graph Autoencoders for Directed Link Prediction[4]



万有引力理论

$$F = \frac{Gm_1m_2}{r^2}, \quad a_{1\to 2} = \frac{F}{m_1} = \frac{Gm_2}{r^2}, \quad a_{2\to 1} = \frac{F}{m_2} = \frac{Gm_1}{r^2}$$

[4]Gravity-Inspired Graph Autoencoders for Directed Link Prediction(CIKM 2019)

Gravity-GAE

基于万有引力的有向边生成模型（Decoder）：

$$\begin{aligned}\hat{A}_{ij} &= \sigma\left(\log a_{i\to j}\right) \\ &= \sigma(\underbrace{\log Gm_j}_{\tilde{m}_j} - \log\|z_i - z_j\|_2^2)\end{aligned}$$

$$p\left(A_{ij}=1 \mid z_i, z_j, \tilde{m}_j\right) = \sigma\left(\tilde{m}_j - \log\|z_i - z_j\|_2^2\right)$$

$$p(A \mid Z, \tilde{M}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p\left(A_{ij} \mid z_i, z_j, \tilde{m}_j\right)$$

Gravity-GAE

基于图神经网络的参数推断（Encoder）：

$$q((Z, \tilde{M}) \mid A, X) = \prod_{i=1}^{n} q\left((z_i, \tilde{m}_i) \mid A, X\right)$$

使用高斯分布建模：

$$q\left((z_i, \tilde{m}_i) \mid A, X\right) = \mathcal{N}\left((z_i, \tilde{m}_i) \mid \mu_i, \mathrm{diag}\left(\sigma_i^2\right)\right)$$

使用图神经网络建模参数：

$$\mu = \mathrm{GCN}_{\mu}(A, X) \text{ and } \log \sigma = \mathrm{GCN}_{\sigma}(A, X)$$

有向图神经网络的实验验证

不同于传统的链接预测任务，有向网络表征学习的链接预测大致分为如下三类：

标准链接预测

随机删除部分边，并抽取部分不存在边的节点对，进行二分类。

有偏链接预测

随机删除部分单向边，并抽取部分不存在边的节点对，进行二分类。

双向链接预测

随机删除双向边中的一条，并抽取部分不存在边的节点对，进行二分类。

Gravity-GAE

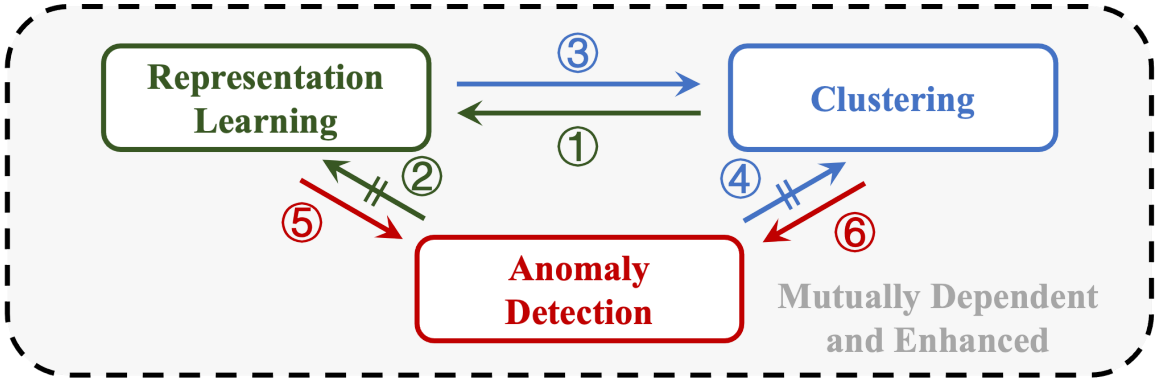
Dataset	Model	Task 1: General Link Prediction		Task 2: B.N.S. Link Prediction		Task 3: Bidirectionality Prediction	
		AUC (in %)	AP (in %)	AUC (in %)	AP (in %)	AUC (in %)	AP (in %)
Cora	*Gravity Graph VAE (ours)*	91.92 ± 0.75	**92.46 ± 0.64**	**83.33 ± 1.11**	**84.50 ± 1.24**	**75.00 ± 2.10**	**73.87 ± 2.82**
	Gravity Graph AE (ours)	87.79 ± 1.07	90.78 ± 0.82	**83.18 ± 1.12**	**84.09 ± 1.16**	**75.57 ± 1.90**	**73.40 ± 2.53**
	Standard Graph VAE	82.79 ± 1.20	86.69 ± 1.08	50.00 ± 0.00	50.00 ± 0.00	58.12 ± 2.62	59.70 ± 2.08
	Standard Graph AE	81.34 ± 1.47	82.10 ± 1.46	50.00 ± 0.00	50.00 ± 0.00	53.07 ± 3.09	54.60 ± 3.13
	Source/Target Graph VAE	85.34 ± 1.29	88.35 ± 0.99	63.00 ± 1.05	64.62 ± 1.37	**75.20 ± 2.62**	**73.86 ± 3.04**
	Source/Target Graph AE	82.67 ± 1.42	83.25 ± 1.51	57.81 ± 2.64	57.66 ± 3.35	65.83 ± 3.87	63.15 ± 4.58
	APP	**93.92 ± 1.01**	**93.26 ± 0.60**	69.20 ± 0.65	67.93 ± 1.09	72.85 ± 1.91	70.97 ± 2.60
	HOPE	80.82 ± 1.63	81.61 ± 1.08	61.84 ± 1.84	63.73 ± 1.12	65.11 ± 1.40	64.24 ± 1.18
	node2vec	79.01 ± 2.00	84.20 ± 1.62	50.00 ± 0.00	50.00 ± 0.00	66.97 ± 1.41	67.61 ± 1.80
Citeseer	*Gravity Graph VAE (ours)*	**87.67 ± 1.07**	**89.79 ± 1.01**	**76.19 ± 1.35**	**79.27 ± 1.24**	**71.61 ± 3.20**	**71.87 ± 3.87**
	Gravity Graph AE (ours)	78.36 ± 1.55	84.75 ± 1.10	**75.32 ± 1.53**	**78.47 ± 1.27**	**71.48 ± 3.64**	**71.50 ± 3.62**
	Standard Graph VAE	78.56 ± 1.43	83.66 ± 1.09	50.00 ± 0.00	50.00 ± 0.00	47.66 ± 3.73	50.31 ± 3.27
	Standard Graph AE	75.23 ± 2.13	75.16 ± 2.04	50.00 ± 0.00	50.00 ± 0.00	45.01 ± 3.75	49.79 ± 3.71
	Source/Target Graph VAE	79.45 ± 1.75	83.66 ± 1.32	57.32 ± 0.92	61.02 ± 1.37	69.67 ± 3.12	67.05 ± 4.10
	Source/Target Graph AE	73.97 ± 3.11	75.03 ± 3.37	56.97 ± 1.33	57.62 ± 2.62	54.88 ± 6.02	55.81 ± 4.93
	APP	**88.70 ± 0.92**	**90.29 ± 0.71**	64.35 ± 0.45	63.70 ± 0.51	64.16 ± 1.90	63.77 ± 3.28
	HOPE	72.91 ± 0.59	71.29 ± 0.52	60.24 ± 0.51	61.28 ± 0.57	52.65 ± 3.05	54.87 ± 1.67
	node2vec	71.02 ± 1.78	77.70 ± 1.22	50.00 ± 0.00	50.00 ± 0.00	61.08 ± 1.88	63.63 ± 2.77
Google	*Gravity Graph VAE (ours)*	**97.84 ± 0.25**	**98.18 ± 0.14**	**88.03 ± 0.25**	**91.04 ± 0.14**	84.69 ± 0.31	84.89 ± 0.30
	Gravity Graph AE (ours)	**97.77 ± 0.10**	**98.43 ± 0.10**	**87.71 ± 0.29**	**90.84 ± 0.16**	**85.82 ± 0.63**	**85.91 ± 0.50**
	Standard Graph VAE	87.14 ± 1.20	88.14 ± 0.98	50.00 ± 0.00	50.00 ± 0.00	40.03 ± 4.98	44.69 ± 3.52
	Standard Graph AE	91.34 ± 1.13	92.61 ± 1.14	50.00 ± 0.00	50.00 ± 0.00	41.35 ± 1.92	41.92 ± 0.81
	Source/Target Graph VAE	96.33 ± 1.04	96.24 ± 1.06	85.30 ± 3.18	84.69 ± 4.42	75.11 ± 2.07	73.63 ± 2.06
	Source/Target Graph AE	**97.76 ± 0.41**	**97.74 ± 0.40**	86.16 ± 2.95	86.26 ± 3.33	82.27 ± 1.29	80.10 ± 1.80
	APP	97.04 ± 0.10	96.97 ± 0.11	83.06 ± 0.46	85.15 ± 0.42	73.43 ± 0.16	68.74 ± 0.19
	HOPE	81.16 ± 0.67	83.02 ± 0.35	74.23 ± 0.80	72.70 ± 0.79	70.45 ± 0.18	70.84 ± 0.22
	node2vec	83.11 ± 0.27	85.79 ± 0.30	50.00 ± 0.00	50.00 ± 0.00	78.99 ± 0.35	76.72 ± 0.53

有向图神经网络

1. 面向有向图的消息传递机制可以直接使用，但是丢失了反向信息，如何设计有向图上的消息传递机制?

$$\mathbf{m}_{i,\leftarrow}^{(k)} = \mathrm{AGG}_{\leftarrow}^{(k)}\left(\left\{(\mathbf{x}_j^{(k-1)}, \mathbf{x}_i^{(k-1)}) : (j,i) \in E\right\}\right)$$
$$\mathbf{m}_{i,\rightarrow}^{(k)} = \mathrm{AGG}_{\rightarrow}^{(k)}\left(\left\{(\mathbf{x}_j^{(k-1)}, \mathbf{x}_i^{(k-1)}) : (i,j) \in E\right\}\right)$$
$$\mathbf{x}_i^{(k)} = \mathrm{COM}^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{m}_{i,\leftarrow}^{(k)}, \mathbf{m}_{i,\rightarrow}^{(k)}\right).$$

2. 如何使用一套表征刻画非对称关系?


Representation Learning
Clustering
Anomaly Detection
③
①
②
④
⑤
⑥
Mutually Dependent and Enhanced

基于聚类假设的数据似然：

$$\begin{aligned}\max \log p(\mathbf{X}|\Theta,\Phi) &= \max \sum_{i=1}^{N} \delta(\mathbf{x}_i) \log p(\mathbf{x}_i|\Theta,\Phi) \\ &= \max \sum_{i=1}^{N} \delta(\mathbf{x}_i) \log \sum_{k=1}^{K} p(\mathbf{x}_i, \mathbf{c}_i = k|\Theta,\Phi),\end{aligned}$$

简单变换：

$$\begin{aligned}p(\mathbf{x}_i|\Theta,\Phi) &= \sum_{k=1}^{K} p(\mathbf{x}_i, \mathbf{c}_i = k|\Theta,\Phi) \\ &= \sum_{k=1}^{K} p(\mathbf{c}_i = k) \cdot p(\mathbf{x}_i|\mathbf{c}_i = k, \Theta, \boldsymbol{\mu}_k, \Sigma_k) \\ &= \sum_{k=1}^{K} \omega_k \cdot p(\mathbf{x}_i|\mathbf{c}_i = k, \Theta, \boldsymbol{\mu}_k, \Sigma_k),\end{aligned}$$

聚类概率分布假设：

$$p(\mathbf{x}_i|\mathbf{c}_i = k, \Theta, \boldsymbol{\mu}_k, \Sigma_k) = \frac{\Gamma(\frac{\nu+1}{2})|\Sigma_k|^{-1/2}}{\Gamma(\frac{\nu}{2})\sqrt{\nu\pi}} \left(1 + \frac{1}{\nu} D_M(\mathbf{z}_i, \boldsymbol{\mu}_k)^2\right)^{-\frac{\nu+1}{2}},$$

似然形式：

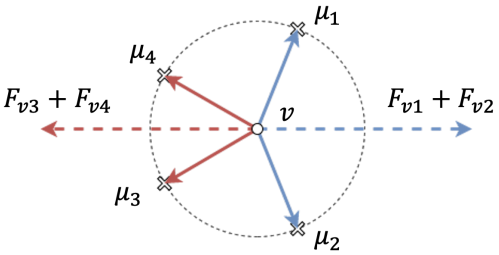
$$p(\mathbf{x}_i|\Theta, \Phi) = \sum_{k=1}^{K} \omega_k \cdot \frac{\pi^{-1} \cdot |\Sigma_k|^{-1/2}}{1 + D_M(\mathbf{z}_i, \boldsymbol{\mu}_k)^2}.$$

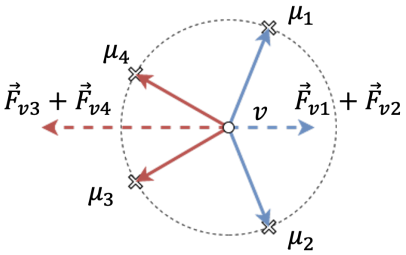
异常检测判定：

$$\delta(\mathbf{x}_i) = \begin{cases} 0, & \text{if } p(\mathbf{x}_i|\Theta, \Phi) \text{ is among the } l \text{ lowest,} \\ 1, & \text{otherwise.} \end{cases}$$

聚类分数：

$$\mathbf{o}_i = \frac{1}{p(\mathbf{x}_i|\Theta, \Phi)} = \frac{1}{\sum_{k=1}^{K} \omega_k \cdot \frac{\pi^{-1} \cdot |\Sigma_k|^{-1/2}}{1 + D_M(\mathbf{z}_i, \boldsymbol{\mu}_k)^2}}.$$



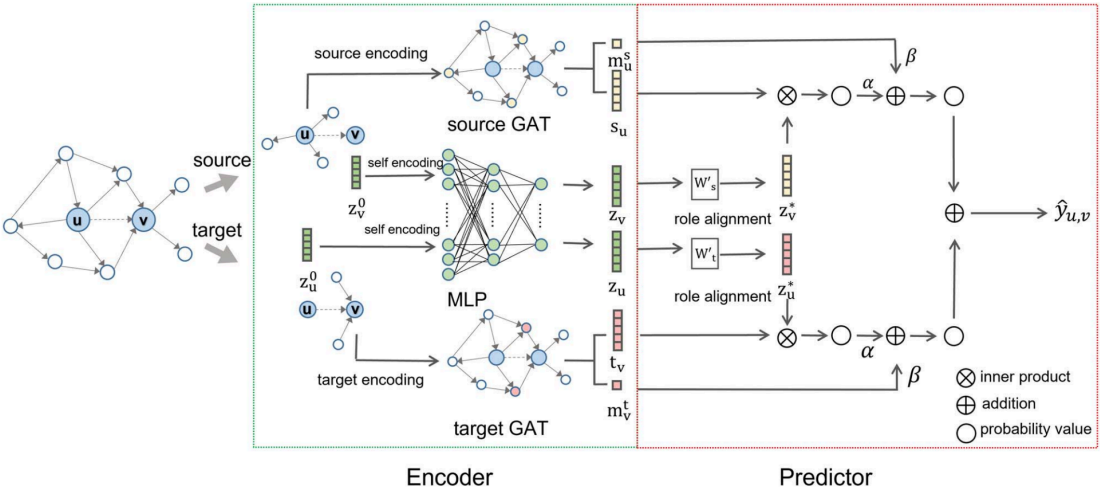


$$\vec{\mathbf{F}}_{i,\text{total}} = \sum_{k=1}^{K} \vec{\mathbf{F}}_{ik}, \quad \text{with } \vec{\mathbf{F}}_{ik} = \frac{G \cdot m_i m_k}{r_{ik}^2} \cdot \vec{\mathbf{r}}_{ik},$$

$$\mathbf{o}_i = \frac{1}{\sum_{k=1}^{K} \widetilde{\mathbf{F}}_{ik}}, \quad \text{with } \widetilde{\mathbf{F}}_{ik} = \frac{\widetilde{G} \cdot \widetilde{m}_i \widetilde{m}_k}{\widetilde{r}_{ik}^2},$$

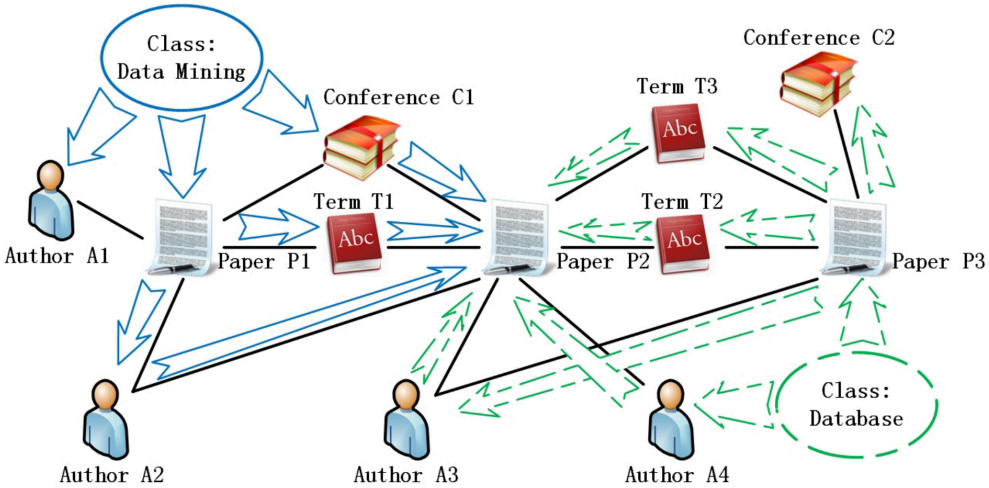
$$\mathbf{o}_i^V = \frac{1}{\| \sum_{k=1}^{K} \widetilde{\mathbf{F}}_{ik} \cdot \vec{\mathbf{r}}_{ik} \|},$$

有向图神经网络应用

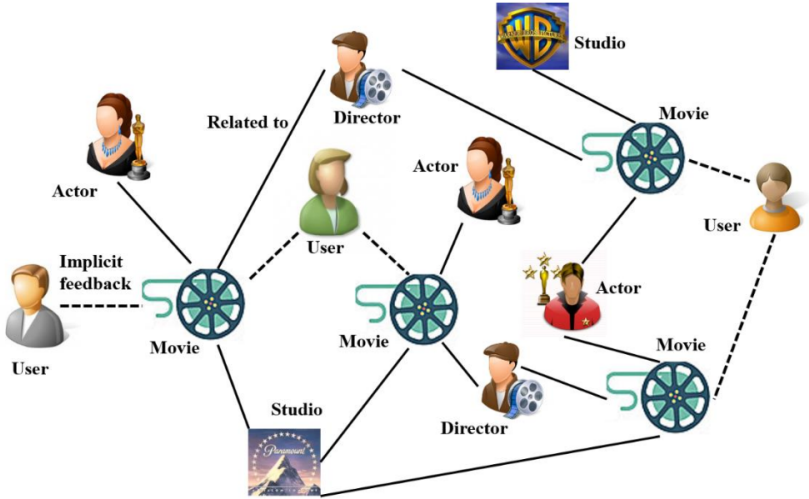


基于有向图神经网络的药物性质预测[5]

异构信息网络


Class:
Data Mining
Conference C1
Conference C2
Term T3
Abc
Term T1
Abc
Term T2
Abc
Author A1
Paper P1
Paper P2
Paper P3
Class:
Database
Author A2
Author A3
Author A4

异构信息网络

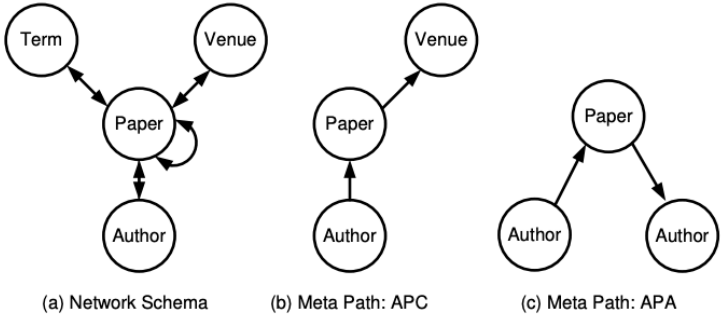


经典异构信息网络

异构信息网络 HIN

异构信息网络的核心：

1. 元路径（Meta-path）：具有语义含义的路径
2. 语义空间（Semantic Space）



Network Schema 和 Meta-path

异构图神经网络

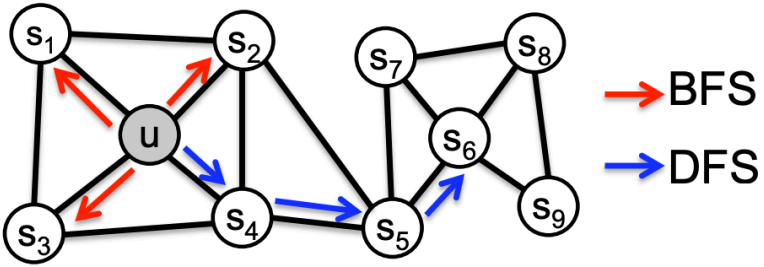
异构图神经网络的难点：

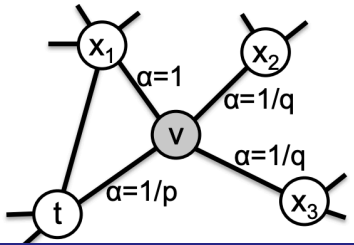
1. 如何定义和选择元路径?
2. 如何在元路径对应的语义空间中学习节点的表征?
3. 如何将不同元路径的表征融合?

发展历程：

1. Node2Vec $\Rightarrow$ Metapath2Vec
2. SDNE $\Rightarrow$ HIN2Vec
3. GAT $\Rightarrow$ HAN
4. GNN $\Rightarrow$ HetGNN

Node2Vec





模型动机

异构网络中节点之间可以通过不同元路径连接，可连接的元路径越多，节点之间的邻近性 (proximity) 越强。

异构 Skip-gram

$$\arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} \log p\left(c_t \mid v; \theta\right)$$

$$p\left(c_t \mid v; \theta\right) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u \in V} e^{X_u \cdot X_v}}$$

异构随机游走

给定元路径: $\mathcal{P}: V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1} \cdots \xrightarrow{R_{l-1}} V_l$，随机游走的转移概率描述为:

$$p\left(v^{i+1} \mid v_t^i, \mathcal{P}\right) = \begin{cases} \frac{1}{\left|N_{t+1}\left(v_t^i\right)\right|} & \left(v^{i+1}, v_t^i\right) \in E, \phi\left(v^{i+1}\right) = t+1 \\ 0 & \left(v^{i+1}, v_t^i\right) \in E, \phi\left(v^{i+1}\right) \neq t+1 \\ 0 & \left(v^{i+1}, v_t^i\right) \notin E \end{cases}$$

Heterogeneous Random Walk 是一种针对异构信息网络的带限制随机游走策略（Restricted Random Walk），其他部分和 Node2Vec 一致。

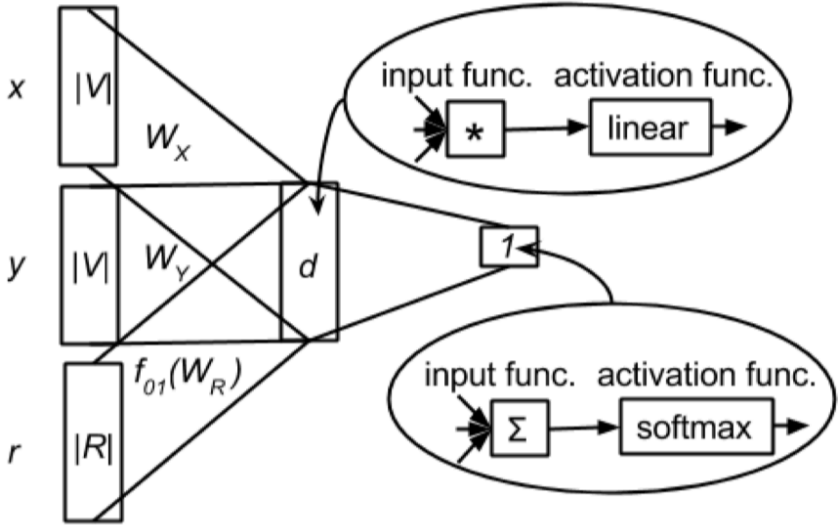
Table 2: Multi-class venue node classification results in AMiner data.

Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.0723	0.1396	0.1905	0.2795	0.3427	0.3911	0.4424	0.4774	0.4955	0.4457
	LINE (1st+2nd)	0.2245	0.4629	0.7011	0.8473	0.8953	0.9203	0.9308	0.9466	0.9410	0.9466
	PTE	0.1702	0.3388	0.6535	0.8304	0.8936	0.9210	0.9352	0.9505	0.9525	0.9489
	metapath2vec	0.3033	0.5247	0.8033	0.8971	0.9406	0.9532	0.9529	0.9701	0.9683	0.9670
	metapath2vec++	0.3090	0.5444	0.8049	0.8995	0.9468	0.9580	0.9561	0.9675	0.9533	0.9503
Micro-F1	DeepWalk/node2vec	0.1701	0.2142	0.2486	0.3266	0.3788	0.4090	0.4630	0.4975	0.5259	0.5286
	LINE (1st+2nd)	0.3000	0.5167	0.7159	0.8457	0.8950	0.9209	0.9333	0.9500	0.9556	0.9571
	PTE	0.2512	0.4267	0.6879	0.8372	0.8950	0.9239	0.9352	0.9550	0.9667	0.9571
	metapath2vec	0.4173	0.5975	0.8327	0.9011	0.9400	0.9522	0.9537	0.9725	0.9815	0.9857
	metapath2vec++	0.4331	0.6192	0.8336	0.9032	0.9463	0.9582	0.9574	0.9700	0.9741	0.9786

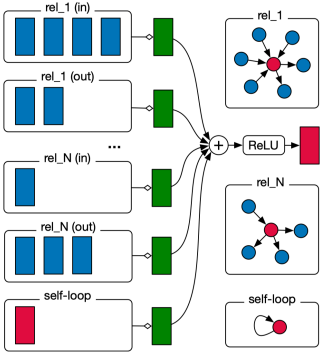
Table 3: Multi-class author node classification results in AMiner data.

Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.7153	0.7222	0.7256	0.7270	0.7273	0.7274	0.7273	0.7271	0.7275	0.7275
	LINE (1st+2nd)	0.8849	0.8886	0.8911	0.8921	0.8926	0.8929	0.8934	0.8936	0.8938	0.8934
	PTE	0.8898	0.8940	0.897	0.8982	0.8987	0.8990	0.8997	0.8999	0.9002	0.9005
	metapath2vec	0.9216	0.9262	0.9292	0.9303	0.9309	0.9314	0.9315	0.9316	0.9319	0.9320
	metapath2vec++	0.9107	0.9156	0.9186	0.9199	0.9204	0.9207	0.9207	0.9208	0.9211	0.9212
Micro-F1	DeepWalk/node2vec	0.7312	0.7372	0.7402	0.7414	0.7418	0.7420	0.7419	0.7420	0.7425	0.7425
	LINE (1st+2nd)	0.8936	0.8969	0.8993	0.9002	0.9007	0.9010	0.9015	0.9016	0.9018	0.9017

HIN2Vec


x
|V|
W_X
y
|V|
W_Y
d
r
|R|
$f_{01}(W_R)$
input func.
activation func.
*
linear
1
input func.
activation func.
Σ
softmax

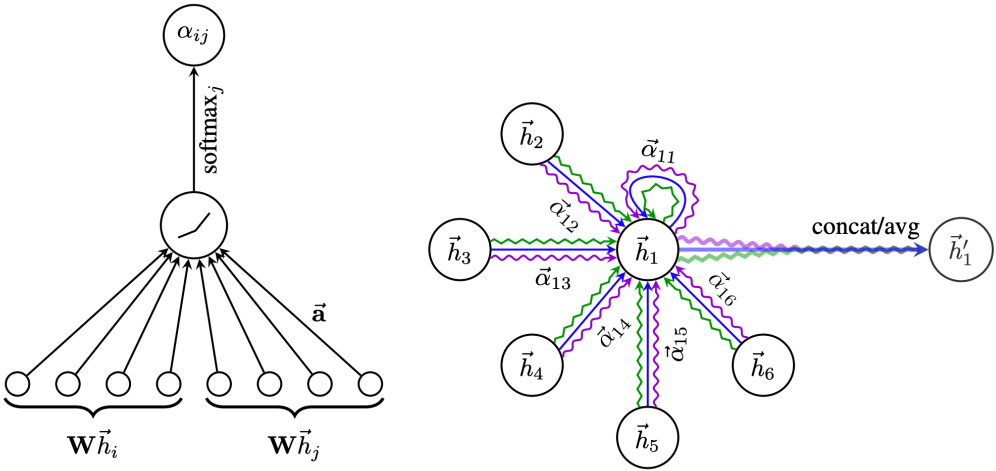
RGCN[7]



$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

[7]Modeling Relational Data with Graph Convolutional Networks(ESWC2018)

Graph Attention Network



Graph Attention Network

Heterogenous Graph Attention Network

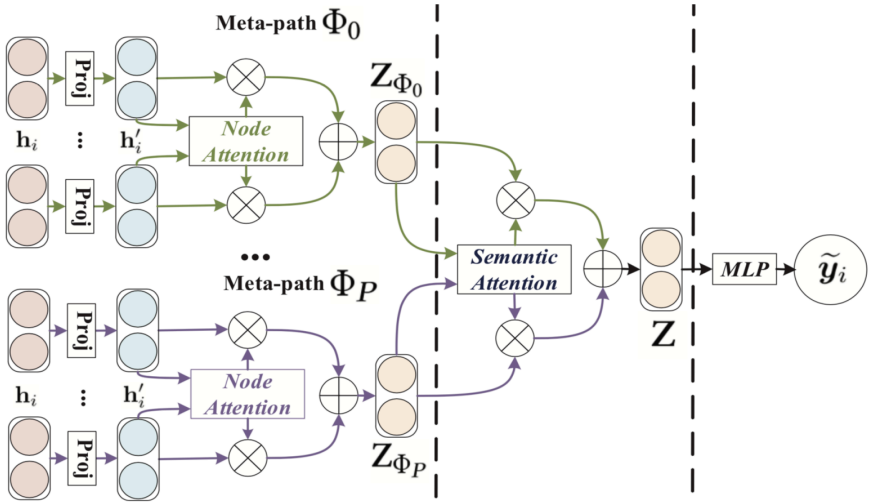
异构图神经网络的特点

1. 不同 meta-path 对应不同语义空间
2. 每个语义空间内节点的邻居和信息均不同
3. 节点的最终表征与每个语义空间均有关

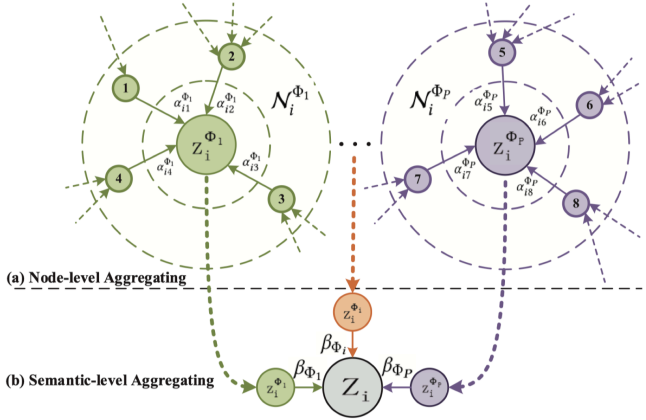
异构图注意力网络

1. 使用 meta-path 异构信息网络投影到多个同构图
2. 在每个同构图内使用注意力网络整合邻居信息
3. 对不同 meta-path 对应的多个同构图使用全局注意力机制

Heterogenous Graph Attention Network



Heterogenous Graph Attention Network



Heterogenous Graph Attention Network

1. Node-level Attention

$$\alpha_{ij}^{\Phi} = \mathrm{softmax}_j\left(e_{ij}^{\Phi}\right) = \frac{\exp\left(\sigma\left(\mathbf{a}_{\Phi}^{\mathrm{T}} \cdot \left[\mathbf{h}_i' \| \mathbf{h}_j'\right]\right)\right)}{\sum_{k \in \mathcal{N}_i^{\Phi}} \exp\left(\sigma\left(\mathbf{a}_{\Phi}^{\mathrm{T}} \cdot \left[\mathbf{h}_i' \| \mathbf{h}_k'\right]\right)\right)}$$

$$\mathbf{z}_i^{\Phi} = \sigma\left(\sum_{j \in \mathcal{N}_i^{\Phi}} \alpha_{ij}^{\Phi} \cdot \mathbf{h}_j'\right)$$

2. Semantic-level Attention

$$w_{\Phi_p} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{q}^{\mathrm{T}} \cdot \tanh\left(\mathbf{W} \cdot \mathbf{z}_i^{\Phi_p} + \mathbf{b}\right)$$

$$\beta_{\Phi_p} = \frac{\exp\left(w_{\Phi_p}\right)}{\sum_{p=1}^{P} \exp\left(w_{\Phi_p}\right)}$$

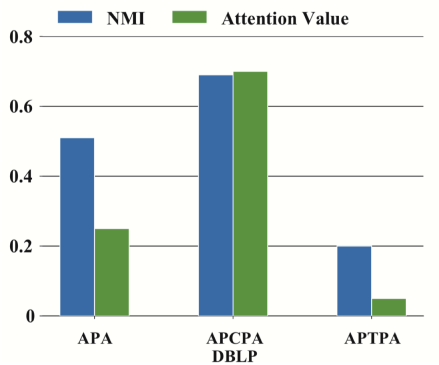
$$\mathbf{Z} = \sum^{P} \beta_{\Phi_p} \cdot \mathbf{Z}_{\Phi_p}$$

Heterogenous Graph Attention Network

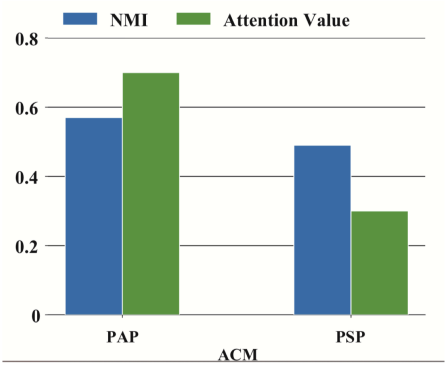
Table 3: Qantitative results (%) on the node classification task.

Datasets	Metrics	Training	DeepWalk	ESim	metapath2vec	HERec	GCN	GAT	HAN$_{nd}$	HAN$_{sem}$	HAN
ACM	Macro-F1	20%	77.25	77.32	65.09	66.17	86.81	86.23	88.15	89.04	**89.40**
		40%	80.47	80.12	69.93	70.89	87.68	87.04	88.41	89.41	**89.79**
		60%	82.55	82.44	71.47	72.38	88.10	87.56	87.91	**90.00**	89.51
		80%	84.17	83.00	73.81	73.92	88.29	87.33	88.48	90.17	**90.63**
	Micro-F1	20%	76.92	76.89	65.00	66.03	86.77	86.01	87.99	88.85	**89.22**
		40%	79.99	79.70	69.75	70.73	87.64	86.79	88.31	89.27	**89.64**
		60%	82.11	82.02	71.29	72.24	88.12	87.40	87.68	**89.85**	89.33
		80%	83.88	82.89	73.69	73.84	88.35	87.11	88.26	89.95	**90.54**
DBLP	Macro-F1	20%	77.43	91.64	90.16	91.68	90.79	90.97	91.17	92.03	**92.24**
		40%	81.02	92.04	90.82	92.16	91.48	91.20	91.46	92.08	**92.40**
		60%	83.67	92.44	91.32	92.80	91.89	90.80	91.78	92.38	**92.80**
		80%	84.81	92.53	91.89	92.34	92.38	91.73	91.80	92.53	**93.08**
	Micro-F1	20%	79.37	92.73	91.53	92.69	91.71	91.96	92.05	92.99	**93.11**
		40%	82.73	93.07	92.03	93.18	92.31	92.16	92.38	93.00	**93.30**
		60%	85.27	93.39	92.48	93.70	92.62	91.84	92.69	93.31	**93.70**
		80%	86.26	93.44	92.80	93.27	93.09	92.55	92.69	93.29	**93.99**
IMDB	Macro-F1	20%	40.72	32.10	41.16	41.65	45.73	49.44	49.78	**50.87**	50.00
		40%	45.19	31.94	44.22	43.86	48.01	50.64	52.11	50.85	**52.71**
		60%	48.13	31.68	45.11	46.27	49.15	51.90	51.73	52.09	**54.24**
		80%	50.35	32.06	45.15	47.64	51.81	52.99	52.66	51.60	**54.38**
	Micro-F1	20%	46.38	35.28	45.65	45.81	49.78	55.28	54.17	55.01	**55.73**
		40%	49.99	35.47	48.24	47.59	51.71	55.91	56.39	55.15	**57.97**
		60%	52.21	35.64	49.09	49.88	52.29	56.44	56.09	56.66	**58.32**
		80%	54.33	35.59	48.81	50.99	54.61	56.97	56.38	56.49	**58.51**

Heterogenous Graph Attention Network



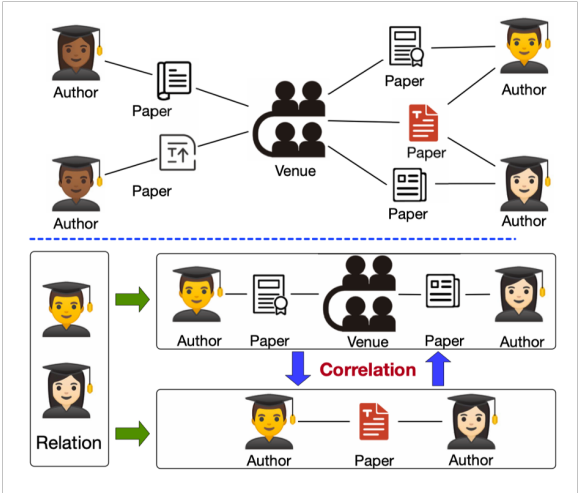
(a) NMI values on DBLP



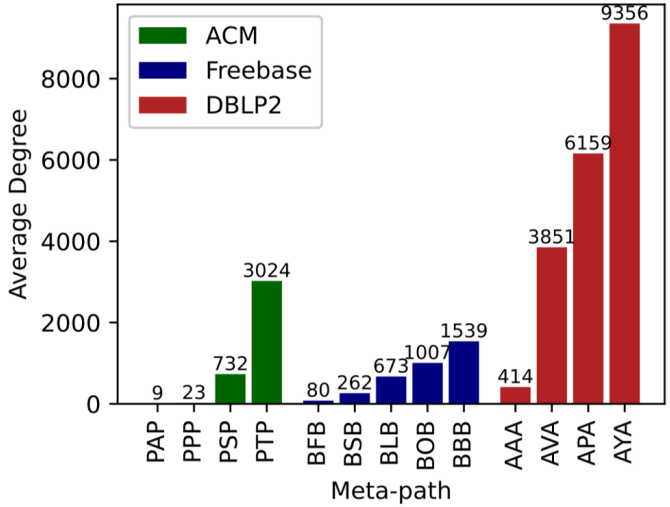
(b) NMI values on ACM

Attention 权重与 meta-path 质量对比

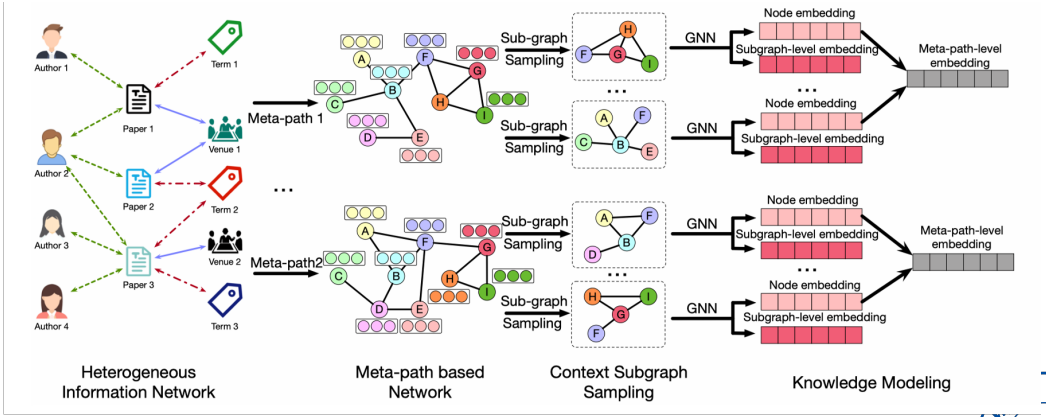
Collaborative Knowledge Distillation for Heterogeneous Information Network Embedding



CKD

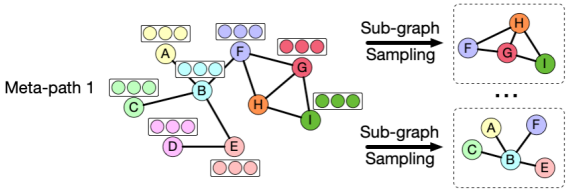

ACM
Freebase
DBLP2
Average Degree
8000
6000
4000
2000
0
9
23
732
3024
80
262
673
1007
1539
414
3851
6159
9356
PAP
PPP
PSP
PTP
BFB
BSB
BLB
BOB
BBB
AAA
AVA
APA
AYA
Meta-path

CKD



基于知识蒸馏的元路径关系挖掘

CKD:Graph Difussion



$$\mathbf{S}^m = \alpha \left(\mathbf{I}_n - (1-\alpha)\mathbf{D}_m^{-1/2}\mathbf{A}^m\mathbf{D}_m^{-1/2} \right)^{-1}$$

CKD:Knowledge Definition

Node-level Knowledge:

$$\mathbf{H}^m = \left(\widetilde{\mathbf{D}}_m^{-\frac{1}{2}} \tilde{\mathbf{A}}^m \widetilde{\mathbf{D}}_m^{-\frac{1}{2}}\right) \mathbf{X}^m \mathbf{W}^m$$

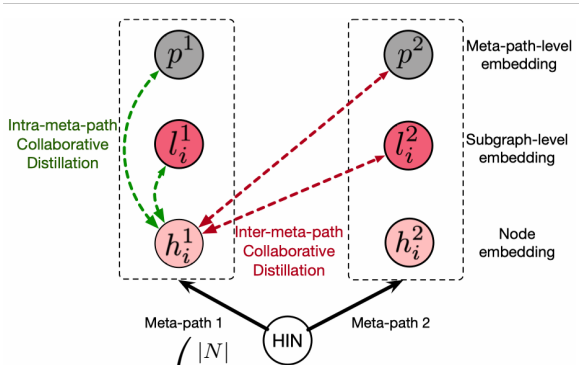
Subgraph-level Knowledge:

$$l_i^m = \mathcal{R}_l\left(G_i^m\right) = \sigma\left(\frac{1}{K}\sum_{j=1}^{K} h_j^m\right)$$

Graph-level Knowledge:

$$p^m = \mathcal{R}_g\left(H^m\right) = \sigma\left(\frac{1}{N}\sum_{i=1}^{N} h_i^m\right)$$

CKD:Optimization



$$\mathcal{L}_{\text{intra}} = -\sum_{m\in\mathcal{M}}\left(\sum_{i}^{|N|}\left(\text{MI}\left(h_i^m, l_i^m\right)+\text{MI}\left(h_i^m, p^m\right)\right)\right)$$

$$\mathcal{L}_{\text{inter}} = -\sum_{i}^{|N|}\left(\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{M},n\neq m}\text{MI}\left(h_i^m, l_i^n\right)+\text{MI}\left(h_i^m, p^n\right)\right)$$

CKD:Results

Table 2: Node classification on six real-world HINs. Bold fonts denote the best performance among all methods. '-' denotes that the method can not be run on our hardware settings. Each method has three lines corresponding to 1/3,1/4,1/5 data for training classifier.

Dataset	ACM		DBLP		ACM2		PubMed		Freebase		DBLP2	
Metric	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
DeepWalk	89.6±0.0	89.6±0.0	91.3±0.0	91.7±0.0	64.8±0.0	75.9±0.0	15.1±0.0	16.8±0.0	**48.9**±0.0	**60.6**±0.0	88.4±0.0	88.3±0.0
	88.8±0.0	88.8±0.0	90.6±0.0	91.0±0.0	64.8±0.0	75.9±0.0	14.7±0.0	16.5±0.0	48.1±0.0	60.1±0.0	88.4±0.0	88.2±0.0
	89.8±0.0	89.8±0.0	90.8±0.0	91.2±0.0	64.6±0.0	76.0±0.0	12.9±0.0	15.7±0.0	**49.3**±0.0	**60.8**±0.0	88.3±0.0	88.1±0.0
Metapath2Vec	91.3±0.3	91.4±0.3	86.3±1.0	87.0±0.9	38.3±1.2	59.0±1.4	13.7±1.2	15.5±1.0	42.2±0.4	54.7±0.2	87.8±0.3	87.6±0.3
	91.7±0.6	91.8±0.6	87.7±1.0	88.3±0.9	38.9±1.1	59.1±1.3	12.4±1.4	14.5±1.2	41.5±1.0	54.6±0.3	88.0±0.2	87.8±0.3
	92.0±0.5	92.1±0.5	89.2±0.5	89.4±0.8	38.8±1.1	59.3±1.5	13.2±1.1	15.2±1.1	41.6±0.3	54.9±0.3	87.9±0.2	87.8±0.1
HIN2Vec	88.5±1.2	88.4±1.3	92.2±0.2	92.5±0.3	23.4±0.6	53.9±0.3	14.8±0.7	18.4±0.5	26.4±0.7	49.3±0.9	86.9±0.4	86.7±0.5
	89.6±1.8	89.8±1.7	91.9±0.2	92.4±0.2	23.7±0.5	54.9±0.6	14.2±0.5	17.8±0.3	25.9±0.4	49.5±0.7	86.6±0.4	86.8±0.3
	89.8±1.6	89.7±1.8	92.5±0.3	93.0±0.2	26.8±0.7	57.4±0.5	14.5±0.8	17.6±0.6	26.0±0.5	49.5±0.8	87.5±0.2	87.3±0.3
HAN	90.4±1.2	90.5±1.2	88.0±0.5	88.5±0.5	59.2±0.9	74.5±0.6	35.1±0.5	37.5±0.3	46.5±0.5	60.1±0.6	88.1±0.6	88.1±0.6
	90.7±1.4	90.8±1.3	87.6±0.7	88.1±0.4	58.7±1.1	74.0±0.8	34.3±0.7	37.1±0.5	46.6±1.1	60.9±0.6	87.5±1.3	87.4±1.4
	90.5±1.0	90.5±1.0	88.4±0.8	88.9±0.8	59.1±0.8	74.5±0.6	35.0±0.8	38.5±0.6	46.7±0.8	60.9±0.4	88.2±0.7	88.2±0.7
HDGI	68.8±2.4	68.9±2.1	74.4±1.0	75.9±1.0	31.5±1.2	57.1±1.2	14.9±0.8	20.3±0.6	-	-	86.8±0.8	87.0±0.8
	68.8±2.2	68.4±2.1	74.5±1.2	75.9±1.1	31.7±1.3	57.2±1.2	15.2±0.7	20.5±0.5	-	-	87.0±0.9	87.2±0.8
	69.8±2.7	69.5±2.8	74.5±1.3	76.0±1.4	31.8±1.4	57.4±1.2	15.4±0.6	20.7±0.4	-	-	87.1±0.9	87.2±0.8
HGT	89.1±0.4	89.3±0.3	50.8±1.0	50.7±1.2	60.9±1.0	75.4±1.2	19.0±0.5	19.9±0.8	-	-	84.1±0.6	84.3±0.6
	89.1±0.5	89.3±0.4	50.9±1.2	51.0±1.1	61.1±1.1	75.7±1.3	20.6±1.9	22.0±1.3	-	-	84.2±0.6	84.4±0.7
	89.2±0.7	89.3±0.7	52.7±0.7	52.8±0.6	61.3±1.2	75.8±1.3	19.4±2.5	20.7±3.7	-	-	84.3±0.9	89.2±0.9
NSHE	90.3±0.3	90.4±0.2	**93.9**±0.1	**94.1**±0.2	62.4±0.6	75.9±0.2	17.1±0.7	22.3±0.9	-	-	-	-
	90.5±0.2	90.6±0.2	**93.8**±0.3	**94.0**±0.3	62.4±0.7	75.9±0.2	17.5±0.8	22.7±0.6	-	-	-	-
	89.7±0.3	89.8±0.3	**93.9**±0.2	**94.1**±0.2	62.5±0.8	76.1±0.2	17.7±0.8	22.9±1.1	-	-	-	-
MAGNN	85.7±0.2	85.7±0.2	87.9±0.3	88.3±0.4	51.0±0.8	70.8±0.4	34.1±1.2	38.3±0.9	47.1±0.6	60.1±0.3	-	-
	87.3±0.4	87.3±0.4	87.5±0.5	88.3±0.2	52.1±0.7	67.8±1.1	36.3±0.6	38.9±0.7	47.6±0.3	60.0±0.5	-	-
	87.9±0.4	88.0±0.4	88.2±0.8	88.9±0.5	53.8±0.6	70.8±0.7	**39.4**±0.7	**42.1**±0.8	47.4±0.7	60.4±0.4	-	-
HeCo	71.0±0.2	71.2±0.1	91.5±0.5	91.8±0.6	57.2±0.8	72.9.0±0.5	16.5±0.5	26.1±1.2	-	-	-	-
	71.2±0.4	71.3±0.3	91.2±0.5	91.4±0.6	56.7±0.9	73.0±0.3	16.8±0.6	25.7±1.1	-	-	-	-
	71.3±0.1	71.3±0.1	91.2±0.4	91.5±0.5	57.5±1.1	72.9±0.7	16.9±0.7	25.9±1.0	-	-	-	-

异构图神经网络是否真的有必要?

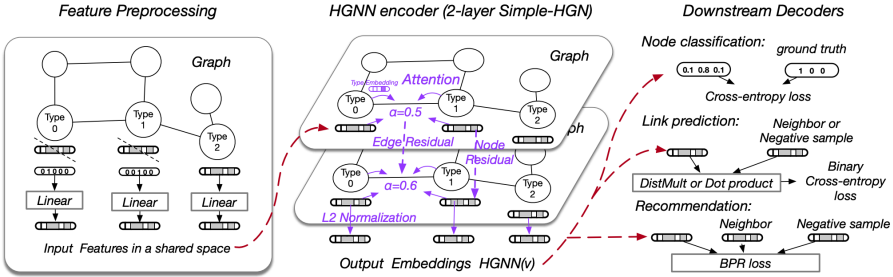


Figure 1: HGB pipeline and Simple-HGN. In this illustration, we assume only the features of Type 2 nodes are kept in the *Feature Preprocessing* period. The purple parts are the improvements over GAT in Simple-HGN.

HGNN 的思考[8]

[8]Are we really making much progress? Revisiting, benchmarking, and refining heterogeneous graph neural networks(KDD 2021)

可学习的边类型表征：

$$\hat{\alpha}_{ij} = \frac{\exp\left(\mathsf{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}_{h_i} \| \mathbf{W}_{h_j} \| \mathbf{W}_{r_r} \boldsymbol{\psi}(\langle i, j \rangle)\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\mathsf{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}_{h_i} \| \mathbf{W}_{h_k} \| \mathbf{W}_{r_r} \boldsymbol{\psi}(\langle i, k \rangle)\right]\right)\right)},$$

节点残差连接：

$$h_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(l)} W^{(l)} h_j^{(l-1)} + W_{\mathsf{res}}^{(l)} h_j^{(l-1)}\right).$$

边残差连接：

$$\alpha_{ij}^{(l)} = (1-\beta)\hat{\alpha}_{ij}^{(l)} + \beta \alpha_{ij}^{(l-1)},$$

异构图神经网络是否真的有必要?

Table 3: Node classification benchmark. Vacant positions ("-") mean that the models run out of memory on large graphs.

	DBLP		IMDB		ACM		Freebase	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
RGCN	91.52±0.50	92.07±0.50	58.85±0.26	62.05±0.15	91.55±0.74	91.41±0.75	46.78±0.77	58.33±1.57
HAN	91.67±0.49	92.05±0.62	57.74±0.96	64.63±0.58	90.89±0.43	90.79±0.43	21.31±1.68	54.77±1.40
GTN	93.52±0.55	93.97±0.54	60.47±0.98	65.14±0.45	91.31±0.70	91.20±0.71	-	-
RSHN	93.34±0.58	93.81±0.55	59.85±3.21	64.22±1.03	90.50±1.51	90.32±1.54	-	-
HetGNN	91.76±0.43	92.33±0.41	48.25±0.67	51.16±0.65	85.91±0.25	86.05±0.25	-	-
MAGNN	93.28±0.51	93.76±0.45	56.49±3.20	64.67±1.67	90.88±0.64	90.77±0.65	-	-
HetSANN	78.55±2.42	80.56±1.50	49.47±1.21	57.68±0.44	90.02±0.35	89.91±0.37	-	-
HGT	93.01±0.23	93.49±0.25	63.00±1.19	67.20±0.57	91.12±0.76	91.00±0.76	29.28±2.52	60.51±1.16
GCN	90.84±0.32	91.47±0.34	57.88±1.18	64.82±0.64	92.17±0.24	92.12±0.23	27.84±3.13	60.23±0.92
GAT	93.83±0.27	93.39±0.30	58.94±1.35	64.86±0.43	92.26±0.94	92.19±0.93	40.74±2.58	65.26±0.80
Simple-HGN	**94.01±0.24**	**94.46±0.22**	**63.53±1.36**	**67.36±0.57**	**93.42±0.44**	**93.35±0.45**	**47.72±1.48**	**66.29±0.45**

仅通过简单的方法即可达到优秀的效果

异构图神经网络论文写作

1. 符号定义清楚
2. Meta-path 等概念定义清楚
3. 画图清晰
4. 如何捕获异构网络的特性?
5. 对比方法公平公正透明
6. 多做 Case Study

动态网络

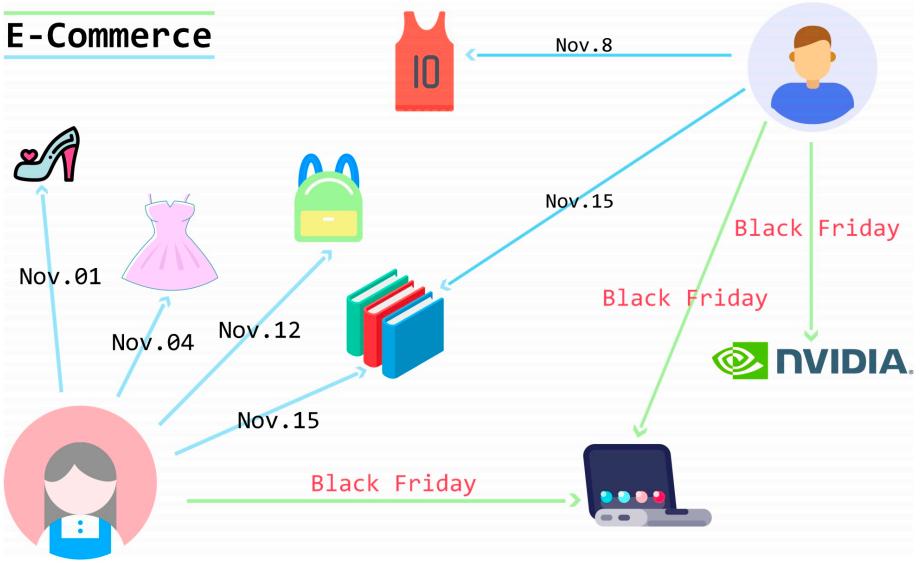
动态网络

- 现实世界中的网络往往都是由不断变化的实体组成的。
- 现有的大部分方法都是将其看作静态的网络，没有考虑动态网络的演化趋势。

常见的动态网络：

- 用户-物品网络
- 货币交易网络
- 社交网络
- ……

用户-物品网络


E-Commerce
Nov.8
Nov.15
Black Friday
Black Friday
Nov.01
Nov.04
Nov.12
Nov.15
Black Friday
NVIDIA

动态网络带来的挑战

捕获时间信息

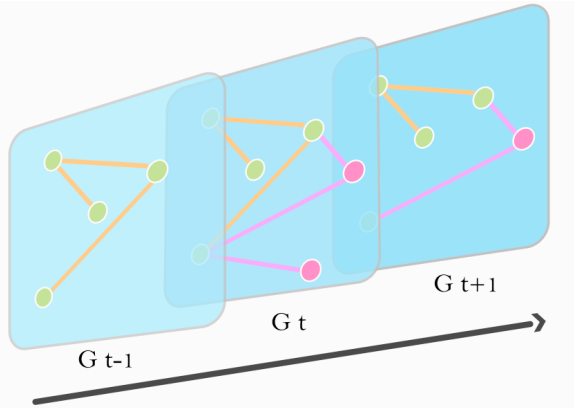
- 边和节点是随时间演化的，邻域聚合会受到时间的约束。
- 学习的节点表示中需要对时间信息进行编码。

高效的表示更新方法

- 一种最简单的更新方法是在每个时间步都应用一次静态网络嵌入方法。然而，这样会消耗大量的时间和计算资源，特别是对于大型的网络。

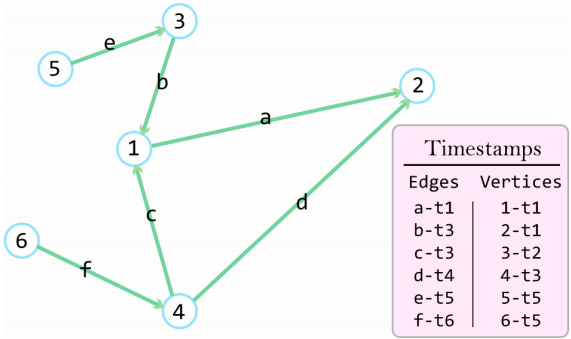
离散模型

- 将动态网络看作一系列的静态网络，连续两个静态网络之间的区别就是发生的变化。



连续模型

- 为每个边和节点指定特定的时间戳，以此来表示发生的变化。



Timestamps

Edges	Vertices
a-t1	1-t1
b-t3	2-t1
c-t3	3-t2
d-t4	4-t3
e-t5	5-t5
f-t6	6-t5

动态网络嵌入方法

根据不同的策略，我们可以将动态网络的嵌入方法分为以下几类：

- 基于矩阵分解的嵌入方法
- 基于Skip-Gram的嵌入方法
- 基于自动编码器的嵌入方法
- 基于神经网络的嵌入方法

基于矩阵分解的嵌入方法

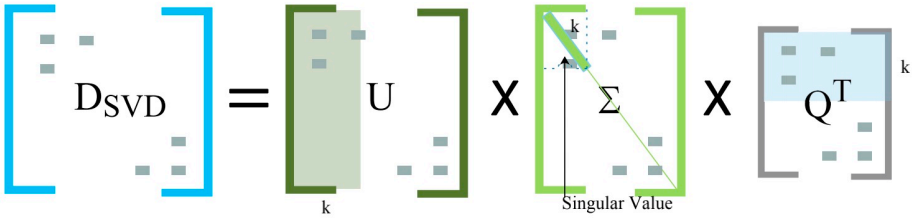
矩阵分解是在网络嵌入领域最普遍的降维方法。其中最具代表性的是奇异值分解（Singular value decomposition，SVD）。

SVD

- 矩阵 $D_{\mathcal{SVD}} \in R^{m \times n}$ 的 SVD 定义为：

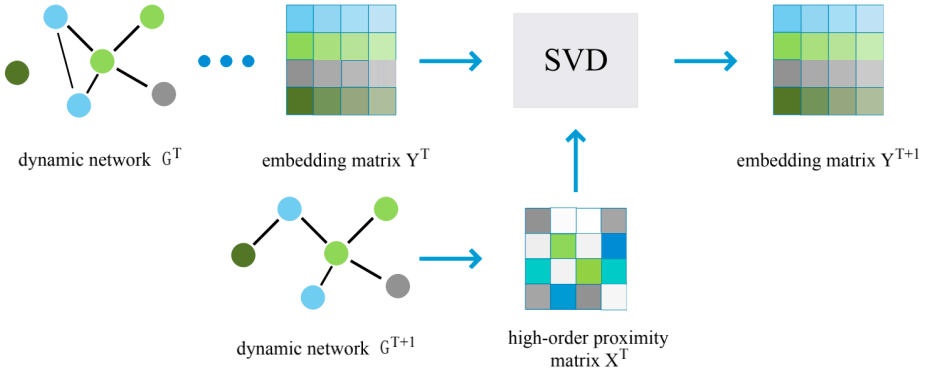
$$D_{\mathcal{SVD}} = U \Sigma Q^T$$

其中 $U \in R^{m \times m}, Q \in R^{n \times n}, \Sigma \in R^{m \times n}$，$\Sigma$ 是一个对角矩阵，主对角线上的每个元素都是一个奇异值。



基于矩阵分解的嵌入方法

从矩阵分解的角度看，网络的动态演化相当于被分解矩阵的不断变化，因此可以根据矩阵扰动理论更新。

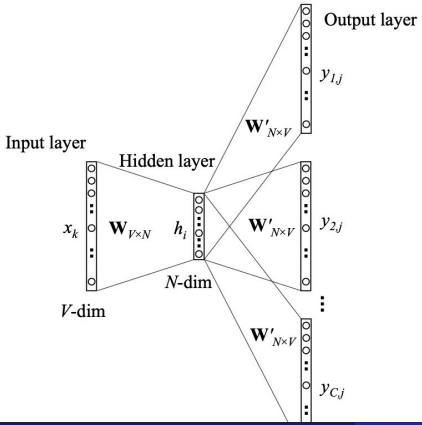


仅适用于网络规模不变的情况

基于 Skip-Gram 的嵌入方法

Skip-Gram

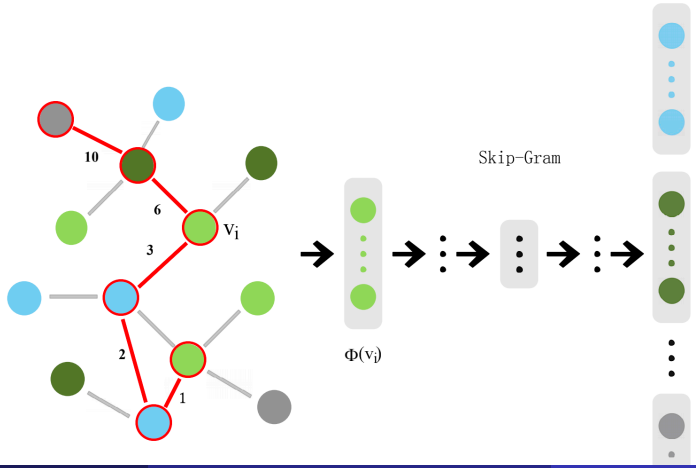
- Word2Vec 的一种经典模型，可以通过输入词来预测上下文。



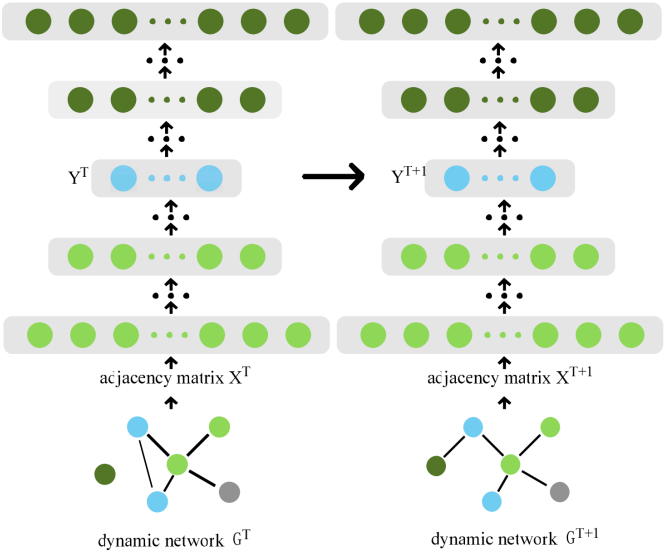
- DeepWalk 提出可以将节点看作为单词，游走序列则对应着句子。
- 由此，大量基于该模型的静态网络嵌入方法被提出。

基于 Skip-Gram 的嵌入方法

- 将动态网络看作连续模型，每条边有相对应的时间戳。
- 此时随机游走中的节点通过一系列时间戳递增的边连接。



基于自动编码器的嵌入方法

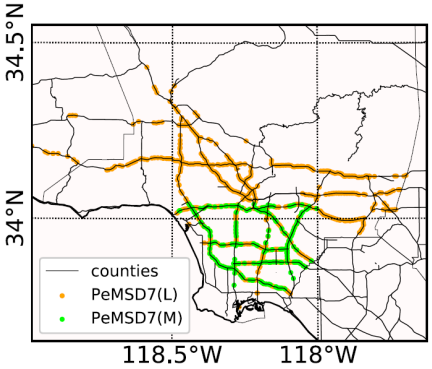


- 离散模型
- 将上一时刻 AE 的权重参数作为下一时刻网络的初始化

STGCN-动态交通流量图预测

交通流量预测

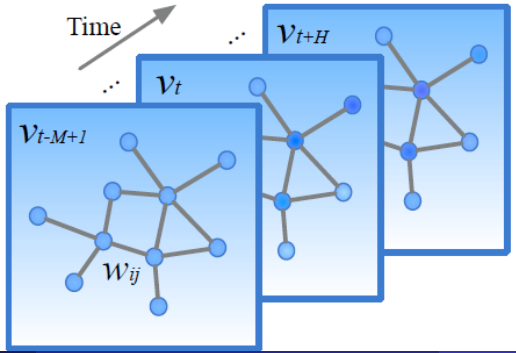
把路口的车流量看作图中节点的属性，道路看作连接节点的边，不同时刻的交通流量网就构成了经典的离散动态图。



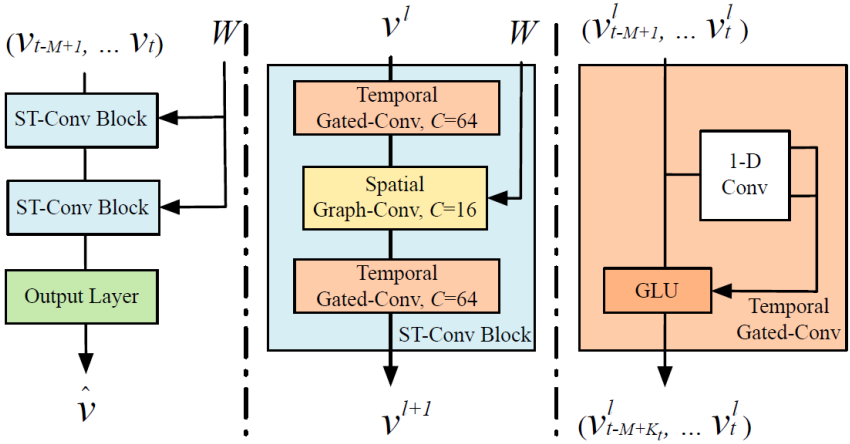
STGCN-动态交通流量图预测

交通流量动态图预测

给定 M 个有序的图 $\mathcal{G}_t = \left(W, X \in R^{M \times n \times C_i}\right)$ ，预测 t+1 时刻的图 $\mathcal{G}_{t+1} = \left(W, X \in R^{n \times C_i}\right)$ 个时间后节点的特征。



STGCN-动态交通流量图预测



STGCN-动态交通流量图预测

时域卷积块

- 输入 $X \in R^{M \times C_i}$, 沿着时间维度进行一维卷积，卷积核 $\Gamma \in R^{K_t \times C_i}$ ，个数为 $2C_o$ ，从而得到 $[PQ] \in R^{(M-K_t+1) \times 2C_o}$, 其中 PQ 指向量在 channel 维度的的前后两部分。
- 然后进行 GLU(gated linear units) 激活: $\Gamma *_\tau X = P \odot \sigma(Q) \in R^{(M-K_t+1) \times C_o}$, $\odot$ 指两个矩阵对应元素相乘的运算。

对于一张完整的时空图: 输入 $X \in R^{M \times n \times C_i}$ ，输出 $Y \in R^{(M-K_t+1) \times n \times C_o}$ 。

离散动态图与连续动态图

连续时间动态图

连续时间动态图由一个按照时间顺序发生的交互序列 $\mathcal{G} = \{(u_i, v_i, t_i)\}_{i=1}^{M}$ 组成，其中 u_i, v_i 表示出发节点和到达节点，$0 \leq t_i \leq t_{i+1}$, $t_i \in \mathbb{R}^+$.

CTDG VS DTDG

相比于离散动态图，连续动态图的主要区别在于网络规模的动态变化，并带来新的困难与挑战：

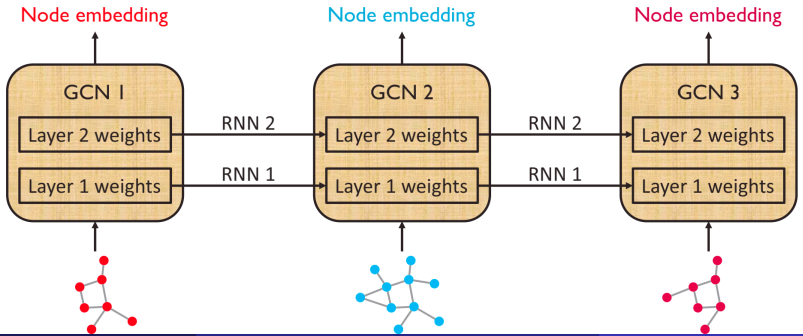
- 无法用统一的邻接矩阵来表示
- 网络规模不断扩大且难以控制
- 新出现节点需严格满足 Inductive 要求

相比于离散时间动态图，连续时间动态图是更一般的动态图形式，连续时间动态图目前是动态图的主流研究方向。

Evolvegcn: Evolving graph convolutional networks for dynamic graphs[9]

GNN 与 RNN 的结合

- 常见的方式是使用 GNN 作为特征提取器，使用 RNN 从提取的节点特征中学习动态。
- 而 EvolveGCN 则着眼于参数，使用 RNN 来演化 GNN 的参数，从演化的网络参数中捕获动态。



EvolveGCN

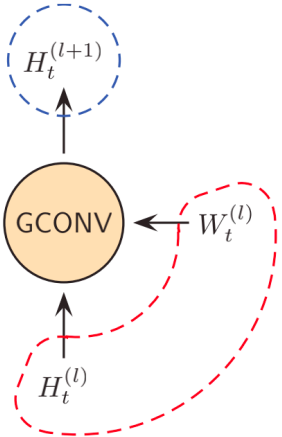
核心思想

- 在离散模型中，对于每个静态网络都需要一个 GCN 模型来训练，从而学到每一层的权重 W。
- 对于动态的场景，当前时刻 GCN 模型的权重与上一时刻 GCN 模型权重有关。
- 因此，如果我们将各个时刻 GCN 每层的模型参数看做一个序列，就可以从 RNN 来学习动态信息。

EvolveGCN

- 在时刻 t，第 l 层以邻接矩阵 A_t 和节点表征矩阵 H_t^l 作为输入，通过权重矩阵 W_t^l 更新节点表征矩阵 $H_t^{(l+1)}$，将其作为输出:

$$H_t^{(t+1)} = \mathbf{GCONV}(A_t, H_t^l, W_t)$$



EvolveGCN

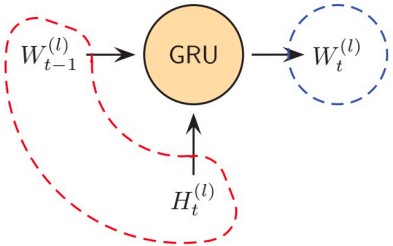
如何将每个时刻 t 模型第 l 的参数 W_t^l 应用于 RNN，EvolveGCN 给出了两个方案：

- EvolveGCN-H：将 W_t^l 作为序列数据的隐藏状态，使用 GRU 来更新 W_t^l。
- EvolveGCN-O：将 W_t^l 作为序列数据的当前输出，也作为下一时刻的输入，通过 LSTM 来建模。

EvolveGCN-H

- 将 W_t^l 作为序列数据的隐藏状态，使用 GRU 来更新 W_t^l。

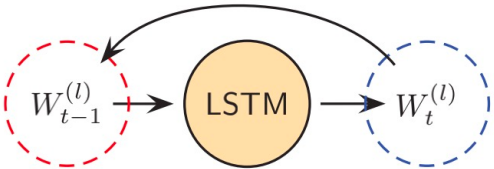
$$\overbrace{\underbrace{W_t^l}_{\text{hidden state}}}^{\text{GCN weights}} = \mathbf{GRU}\Big(\overbrace{\underbrace{H_t^l}_{\text{input}}}^{\text{node embeddings}} + \overbrace{\underbrace{W_{t-1}^l}_{\text{hidden state}}}^{\text{GCN weights}}\Big)$$



EvolveGCN-O

- 将 W_t^l 作为序列数据的当前输出，也作为下一时刻的输入，通过 LSTM 来建模。

$$\overbrace{\underbrace{W_t^l}_{\text{output}}}^{\text{GCN weights}} = \mathbf{LSTM}(\overbrace{\underbrace{W_{t-1}^l}_{\text{input}}}^{\text{GCN weights}})$$



EGCU

将上述 GCN 单元和 RNN 单元结合，可以组合为新的演化图卷积单元（Evolving Graph Convolution Unit，EGCU）。

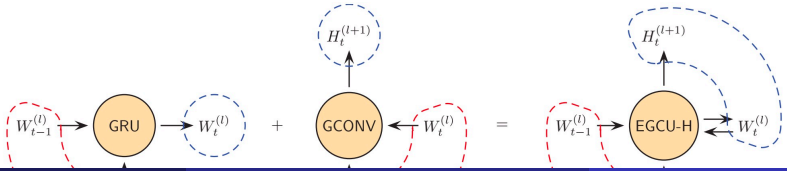
- EGCU-H：

1: **function** $[H_t^{(l+1)}, W_t^{(l)}] = \text{EGCU-H}(A_t, H_t^{(l)}, W_{t-1}^{(l)})$

2: $\quad W_t^{(l)} = \text{GRU}(H_t^{(l)}, W_{t-1}^{(l)})$

3: $\quad H_t^{(l+1)} = \text{GCONV}(A_t, H_t^{(l)}, W_t^{(l)})$

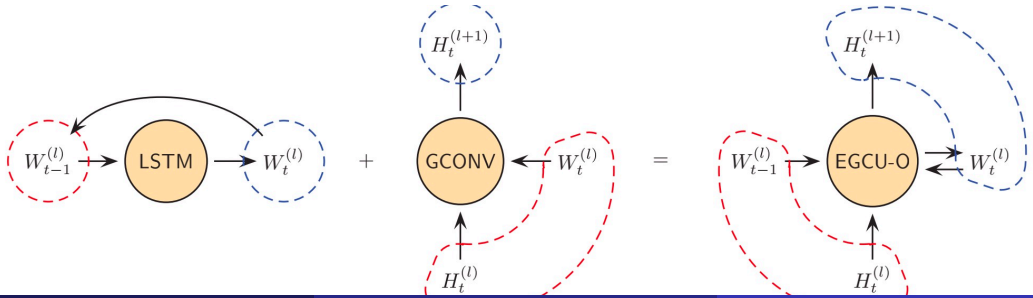
4: **end function**



EGCU

- EGCU-O:

1: **function** $[H_t^{(l+1)}, W_t^{(l)}] = \text{EGCU-O}(A_t, H_t^{(l)}, W_{t-1}^{(l)})$
2: $\quad W_t^{(l)} = \text{LSTM}(W_{t-1}^{(l)})$
3: $\quad H_t^{(l+1)} = \text{GCONV}(A_t, H_t^{(l)}, W_t^{(l)})$
4: **end function**



不同方案的抉择

两种方案

EvolveGCN-H:

$$\overbrace{\underbrace{W_t^l}_{\text{hidden state}}}^{\text{GCN weights}} = \mathbf{GRU}\Big(\overbrace{\underbrace{H_t^l}_{\text{input}}}^{\text{node embeddings}} + \overbrace{\underbrace{W_{t-1}^l}_{\text{hidden state}}}^{\text{GCN weights}}\Big)$$

EvolveGCN-O:

$$\overbrace{\underbrace{W_t^l}_{\text{output}}}^{\text{GCN weights}} = \mathbf{LSTM}\Big(\overbrace{\underbrace{W_{t-1}^l}_{\text{input}}}^{\text{GCN weights}}\Big)$$

- 当节点属性比较丰富时，H 方案会更有效，因为它的 RNN 中包含了额外的节点表征输入。
- 当网络的结构更加重要时，O 方案更加关注结构的变化，效果相对较好。

基于随机游走的方法——CTDNE [10]

时序游走

给定动态图 $\mathcal{G} = \{(u_i, v_i, t_i)\}_{i=1}^{M}$，$\mathcal{G}$ 上从 t_m 时刻的 w_m 节点出发，长度为 m 的随机游走 W 定义为：

$$W = ((w_m, t_m), (w_{m-1}, t_{m-1}), ..., (w_0, t_0))$$
$$\text{s.t.} \quad t_m > t_{m-1} > ... > t_0, (w_i, w_{i-1}, t_{i-1}) \in \mathcal{G}$$

动态图中的时序游走从时间戳靠后的节点开始，逆时间的沿着的时序边游走，到达时间戳靠前的节点。

[10] Continuous-Time Dynamic Network Embeddings (WWW 2018)

CTDNE——选择初始边

选择时序游走的起始边

$\mathcal{G}$ 上的任意一条边 $e = (u, v, t) \in \mathcal{G}$ 作为起始边的概率为：

- 无偏：均匀采样

$$Pr(e) = \frac{1}{|\mathcal{G}|}$$

- 有偏：时间戳靠后的边采样概率更大

$$Pr(e) = \frac{\exp(t - t_{min})}{\sum_{e' \in \mathcal{G}} \exp(t' - t_{min})}$$

CTDNE——训练

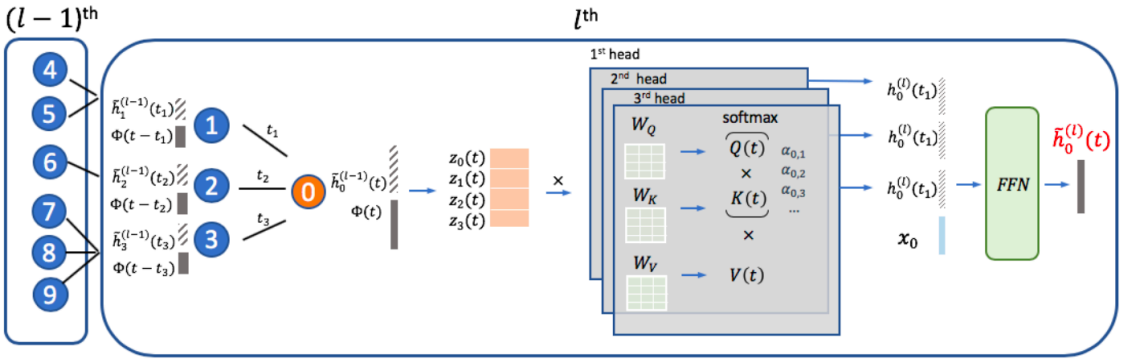
得到时序游走序列后，CTDNE 采用 Skip-Gram 的思想，最大化窗口内节点嵌入共现的概率：

$$\max_{f} \log Pr(W = \{v_{i-\omega}, ..., v_{i+\omega}\}\ v_i | f(v_i))$$

其中 ω 为窗口大小，$f(v_i)$ 表示 v_i 的嵌入向量。上式经过分解得到：

$$Pr(W = |f(v_i)) = \prod_{v_{i+k} \in W} Pr(v_{i+k} | f(v_i))$$

基于消息传递的方法——TGAT [11]



[11] Inductive Representation Learning On Temporal Graphs (ICLR 2020)

TGAT———输入层

TGAT 将消息传递机制拓展到了动态图，逐层采样并聚合历史邻居的节点表征。

TGAT 输入层

对于节点 v_0 在 t 时刻第 l 层的表征，TGAT 考虑它自身和历史邻居 $\mathcal{N}(v_0;t)=\{(v_1,t_1)...,(v_N,t_N)\}$ 第 $(l-1)$ 层的表征作为输入：

$$\mathbf{Z}(t)=[\tilde{h}_0^{(l-1)}(t)||\Phi(0),\tilde{h}_1^{(l-1)}(t)||\Phi(t-t_1),...,\tilde{h}_N^{(l-1)(t)}||\Phi(t-t_N)]$$

其中 $\Phi(\cdot):\mathbb{R}^1\to\mathbb{R}^d$ 是时间表征函数，将时间差转化成向量。

TGAT——前向过程

TGAT 采用注意力机制。首先得到 query, key, value:

$$\mathbf{q}(t) = [\mathbf{Z}(t)]_0 \mathbf{W}_Q, \mathbf{K}(t) = [\mathbf{Z}(t)]_{1:N} \mathbf{W}_Q, \mathbf{V}(t) = [\mathbf{Z}(t)]_{1:N} \mathbf{W}_V$$

经过局部注意力层：

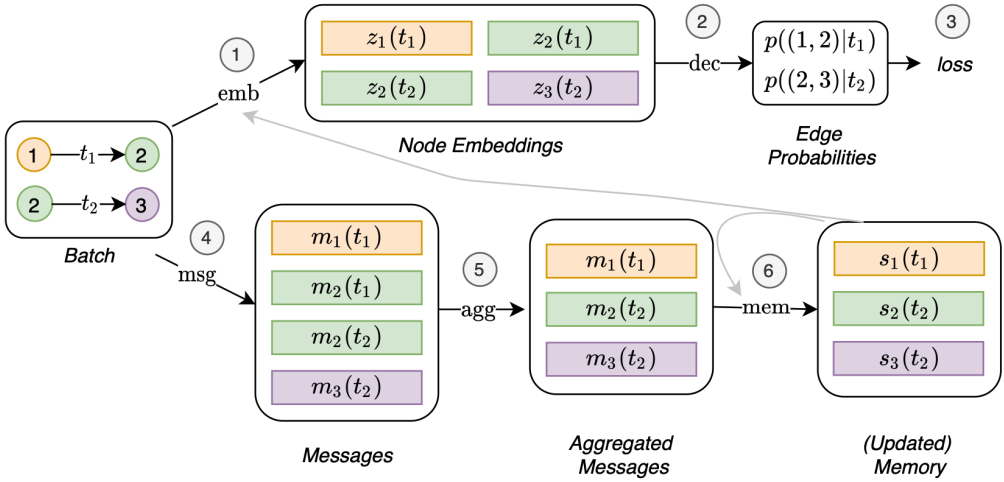
$$\mathbf{h}(t) = \mathrm{Attn}(\mathbf{q}(t), \mathbf{K}(t), \mathbf{V}(t)) \in \mathbb{R}^{d_h},$$

经过全连接层，得到下一层的表征：

$$\tilde{\mathbf{h}}_0^{(l)}(t) = \mathrm{FFN}(\mathbf{h}(t) || \mathbf{x}_0)$$

由于历史邻居的数量较大，TGAT 需要对邻居进行采样。

基于记忆机制的方法-TGN [12]



TGN 在 TGAT 的基础上进行了改进，增加了 memory 机制，用于维护长期记忆信息。TGN 为每个节点 i 分配了一个记忆向量 $s_i(t)$，每当 i 有新的交互（消息）发生，$s_i(t)$ 会更新。

TGN

消息函数

当交互事件 (i, j, t, e_{ij}) 发生时，TGN 针对节点 i, j 产生两条消息：

$$\mathbf{m}_i(t) = \mathrm{msg}(\mathbf{s}_i(t^-), \mathbf{s}_j(t^-), \Delta t, \mathbf{e}_{ij}(t))$$
$$\mathbf{m}_j(t) = \mathrm{msg}(\mathbf{s}_j(t^-), \mathbf{s}_i(t^-), \Delta t, \mathbf{e}_{ij}(t))$$

其中 msg 表示恒等函数，$\Delta(t)$ 表示与上一次 memory 更新的时间差

记忆更新

节点 i 的记忆利用消息进行更新：

$$\mathbf{s}_i(t) = \mathrm{mem}(\mathbf{m}_i(t), \mathbf{s}_i(t^-))$$

mem 为记忆更新函数，实现为 LSTM 或 GRU.

TGN——节点嵌入

节点嵌入

当计算节点 i 在 t 时刻的表征时，TGN 聚合历史邻居的信息：

$$\mathbf{z}_i(t) = \sum_{j \in \eta_i^K(t)} h(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathbf{s}_{ij}, \mathbf{x}_i, \mathbf{x}_j)$$

其中 $\eta_i^K(t)$ 为 K 跳历史邻居。h 可以有多种形式：

- 时间映射：$\mathbf{z}_i(t) = (1 + \Delta t \mathbf{w}) \circ \mathbf{s}_i(t)$，$\mathbf{w}$ 为可学向量。将 memory 按照时间差进行线性映射，即为 JODIE [a]。
- 时序图注意力网络：此即为 K 层 TGAT 网络。

[a] Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks (KDD 2019)

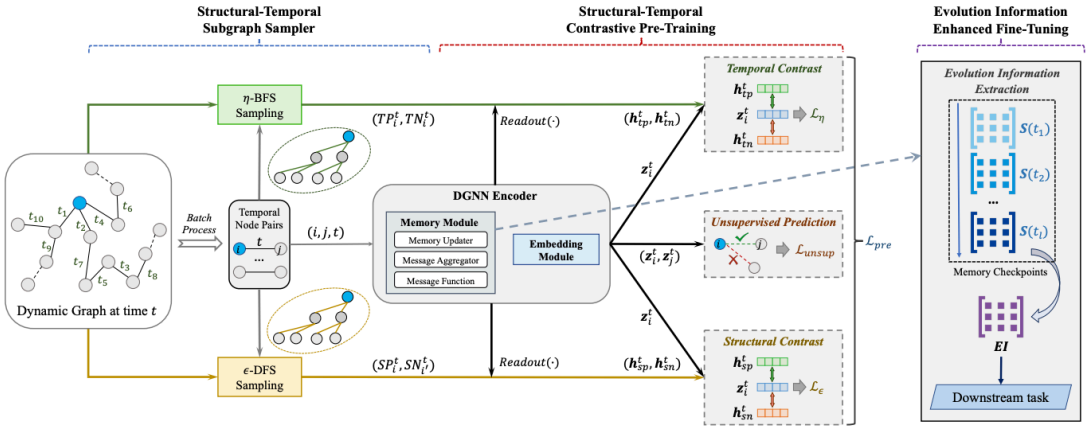
动态图预训练

- 目前已有许多工作关注到静态图上的预训练，然而在真实工业应用中，对于动态性质的建模具有重要意义，如在推荐系统中，用户兴趣的演变具有动态性。
- 而现有工作较少的关注到动态图上的预训练方式的设计。

动态图预训练需要关注到：

- 下游任务的泛化性
- 长短期建模能力

Overall Framework of CPDG



CPDG 主要包括三个模块：

(1) Structural-Temporal Subgraph Sampler;

(2) Structural-Temporal Contrastive Pre-Training;

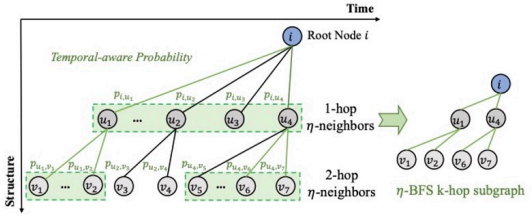
Structural-Temporal Subgraph Sampler

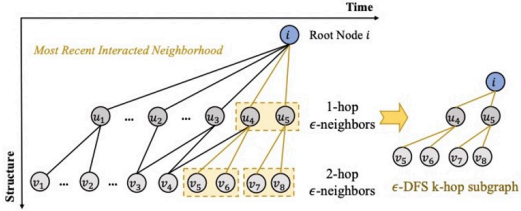
核心思想

- 为时间-结构对比预训练提供训练样本：时间侧 (η-BFS 采样), 结构侧 (ϵ-DFS 采样)
- η-BFS 采样：按照时间采样函数，抽取不同的时间偏好子图
- ϵ-DFS 采样：固定时间维度最近邻，抽取结构偏好的子图

采样器示例

- η-BFS 采样：为需要采样的各个节点根据时间函数赋予采样概率，进行概率采样（左图为 η-BFS 采样示例）。
- ϵ-DFS 采样：固定采各阶子图的最近时间节点，进行结构偏好的采样（右图为 ϵ-DFS 采样示例）。





Temporal Contrast 核心思想

- 对于节点 i, 在某一时刻 t，以正比于时间的采样概率作为正样本采样，以反比于时间的采样概率作为负样本采样，通过 triplet margin loss 作为目标损失进行学习。
- 因此，如果我们全局看各个时刻 t 的对比学习，Temporal Contrast 学习了各时刻的 short-term 演变，结合动态图神经网络本身 memory 机制对于 long-term 模式的学习，从而完成对于 long-short trem 的建模。

$$\mathbf{h}_{tp}^t = \text{Readout}\left(\mathbf{s}_u^t, u \in \mathcal{TP}_i^t\right),$$
$$\mathbf{h}_{tn}^t = \text{Readout}\left(\mathbf{s}_v^t, v \in \mathcal{TN}_i^t\right),$$

$$\mathcal{L}_\eta = \frac{1}{|\mathcal{V}^t|} \sum_{i \in \mathcal{V}^t} \mathbb{E}\left\{\max\{d\left(\mathbf{z}_i^t, \mathbf{h}_{tp}^t\right) - d\left(\mathbf{z}_i^t, \mathbf{h}_{tn}^t\right) + \alpha, 0\}\right\}.$$

Structural Contrast 核心思想

- 对于节点 i, 在某一时刻 t，固定最近时间进行各结构偏好子图采样，以其自身的子图作为正样本，随机采样另一其他节点 i' 的子图作为负样本。
- 因此，Structural Contrast 侧重于结构模式的学习，结合 Temporal Contrast 共同完成时间-结构模式的学习。

$$\mathbf{h}_{sp}^{t} = \text{Readout}\left(\mathbf{s}_{u}^{t}, u \in \mathcal{SP}_{i}^{t}\right),$$
$$\mathbf{h}_{sn}^{t} = \text{Readout}\left(\mathbf{s}_{v}^{t}, v \in \mathcal{SN}_{i'}^{t}\right),$$

$$\mathcal{L}_{\epsilon} = \frac{1}{|\mathcal{V}^{t}|} \sum_{i \in \mathcal{V}^{t}} \mathbb{E}\left\{\max\{d\left(\mathbf{z}_{i}^{t}, \mathbf{h}_{sp}^{t}\right) - d\left(\mathbf{z}_{i}^{t}, \mathbf{h}_{sn}^{t}\right) + \alpha, 0\}\right\}.$$

Evolution Information Enhanced Fine-Tuning

核心思想

- 由于预训练的动态图神经网络的 memory 中在预训练过程中存储了丰富的图演化信息（Evolution Information），对于预训练过程中已经出现过的节点，我们可以基于此信息辅助其下游微调过程。
- 具体地，我们可以在预训练过程存储 memory checkpoints，将这些 checkpoints 提取为表示向量最终辅助下游节点 embedding 的学习。

$$\mathbf{EI} = f_{EI}\left([\mathbf{S}^1, ..., \mathbf{S}^l, ..., \mathbf{S}^L]\right),$$

$$\mathbf{Z}^{EIE} = [\mathbf{Z}^{\text{down}} \parallel \text{MLP}(\mathbf{EI})].$$

课程总结

1. 上节课回顾
2. 研究背景
3. 有向图神经网络
4. 异构图神经网络
5. 动态图神经网络

References I

Khosla, M., Leonhardt, J., Nejdl, W., and Anand, A. (2019).
Node representation learning for directed graphs.
In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 395–411. Springer.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016).
Asymmetric transitivity preserving graph embedding.
In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114.

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., and Leiserson, C. (2020).
Evolvegcn: Evolving graph convolutional networks for dynamic graphs.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5363–5370.

References II

Salha, G., Limnios, S., Hennequin, R., Tran, V.-A., and Vazirgiannis, M. (2019).
Gravity-inspired graph autoencoders for directed link prediction.
In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 589–598.

Zhou, C., Liu, Y., Liu, X., Liu, Z., and Gao, J. (2017).
Scalable graph embedding for asymmetric proximity.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Zhou, S., Wang, X., Ester, M., Li, B., Ye, C., Zhang, Z., Wang, C., and Bu, J. (2021).
Direction-aware user recommendation based on asymmetric network embedding.
ACM Transactions on Information Systems (TOIS), 40(2):1–23.