# 图神经网络导论

## 富信息图神经网络

授课教师：周晟

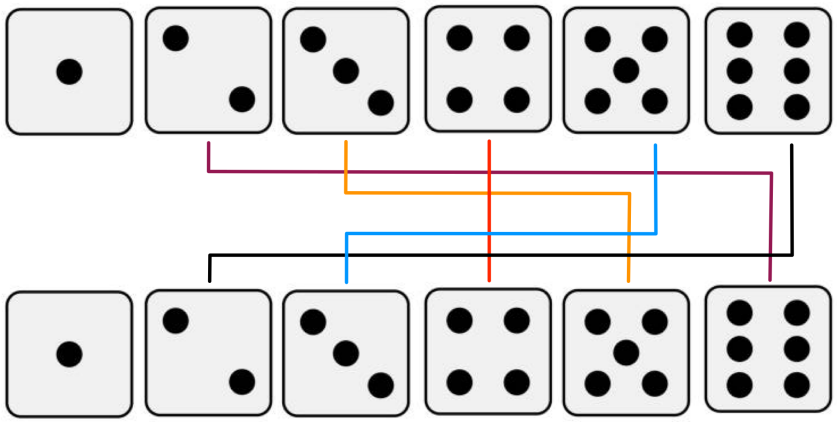
浙江大学　软件学院

2022.11

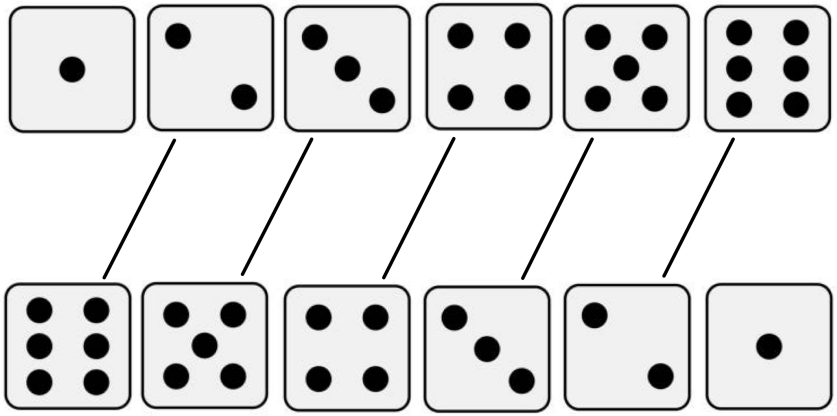
# 课程内容



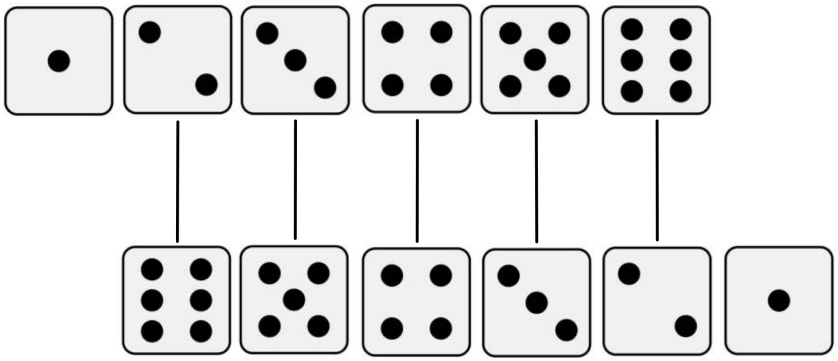
# 生活中的卷积



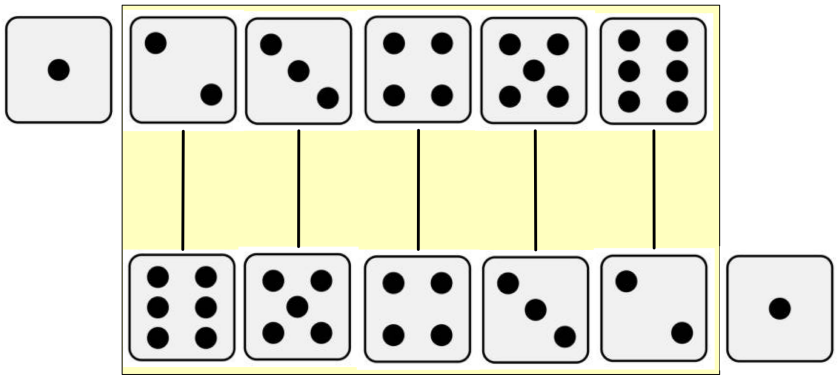
解



卷



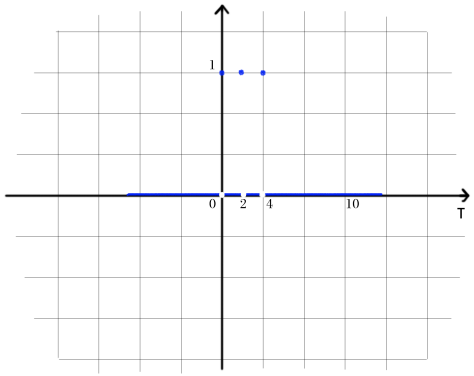
移



积

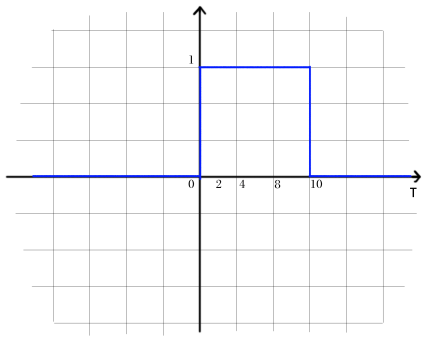
# 生活中的卷积


肥宅快乐问题




间隔的快乐

肥宅快乐问题




连续的快乐

# 数学上的卷积

## 卷积的定义

卷积是一种定义在两个函数 $f, g$ 上的数学操作 $(f * g)(n)$：

1. 连续卷积:

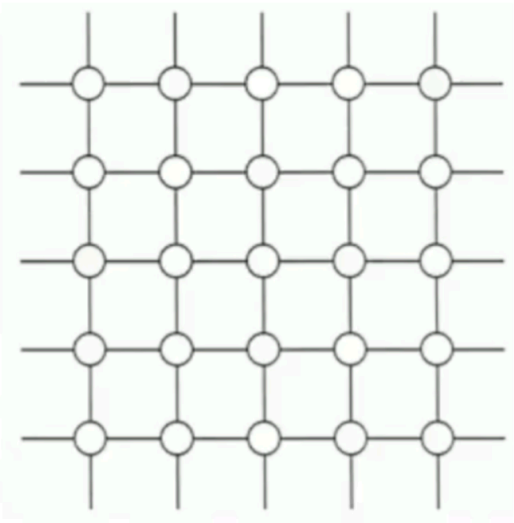
$$(f * g)(n) = \int_{-\infty}^{\infty} f(\tau)g(n-\tau)d\tau$$

2. 离散卷积:

$$(f * g)(n) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(n-\tau)$$

卷积的三个基本操作：翻转，滑动，叠加

# 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）
2. 节点邻居没有固定个数（难以定义统一的特征提取函数）

# 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：

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

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

# 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

# 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. 传统图神经网络难以充分建模富信息网络特点
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

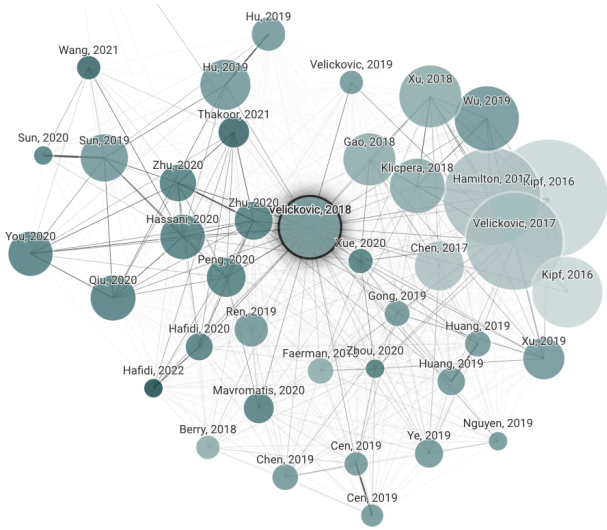
1. 有向图
2. 异构图
3. 动态图

# 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. Convolutional 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 | $\mathbf{I}$ | $\mathbf{A} \cdot \mathbf{D} \cdot \mathbf{A}$ |

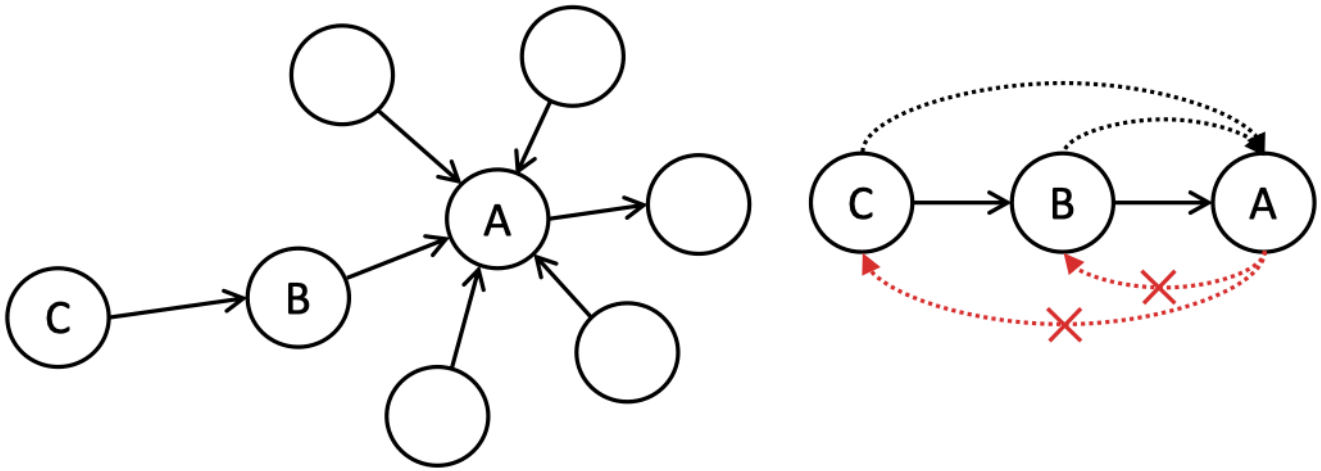
使用矩阵分解方法，使得每个节点有两套表征。

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

# Scalable Graph Embedding for Asymmetric Proximity (APP)

## 研究动机

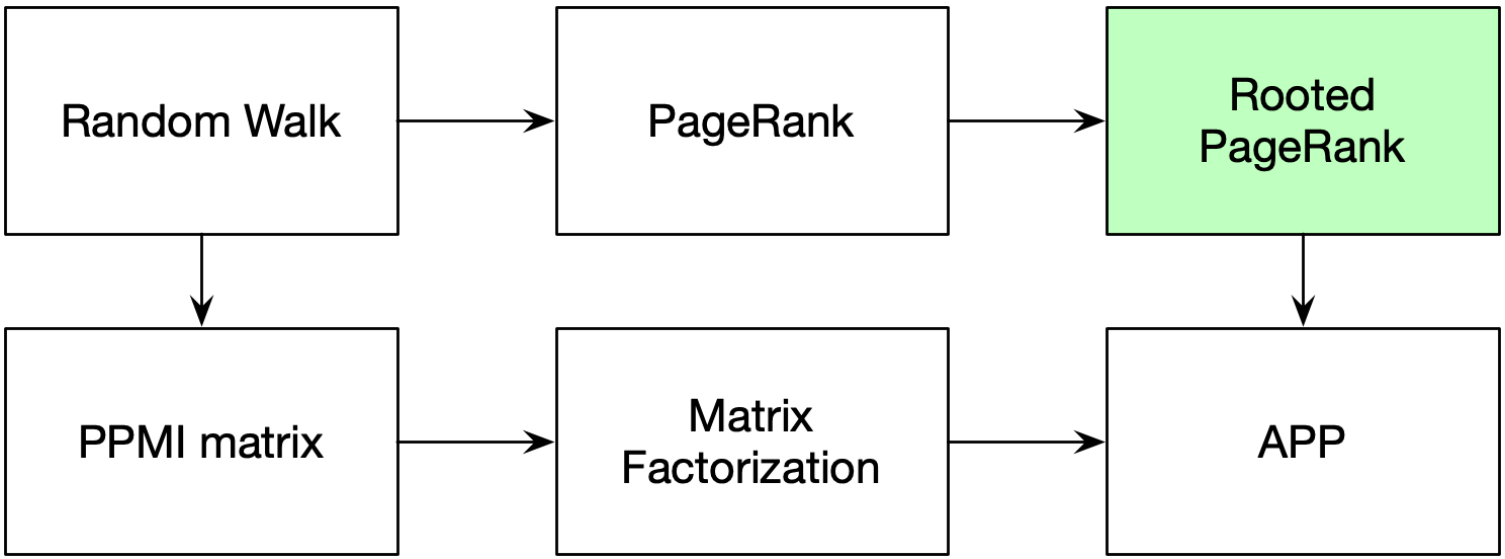
图表征学习的核心是节点之间的临近性（Proximity），然而有向图中的临近性是非对称的（Asymmetric）




Asymmetric Proximity[2]

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

# APP




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

# APP

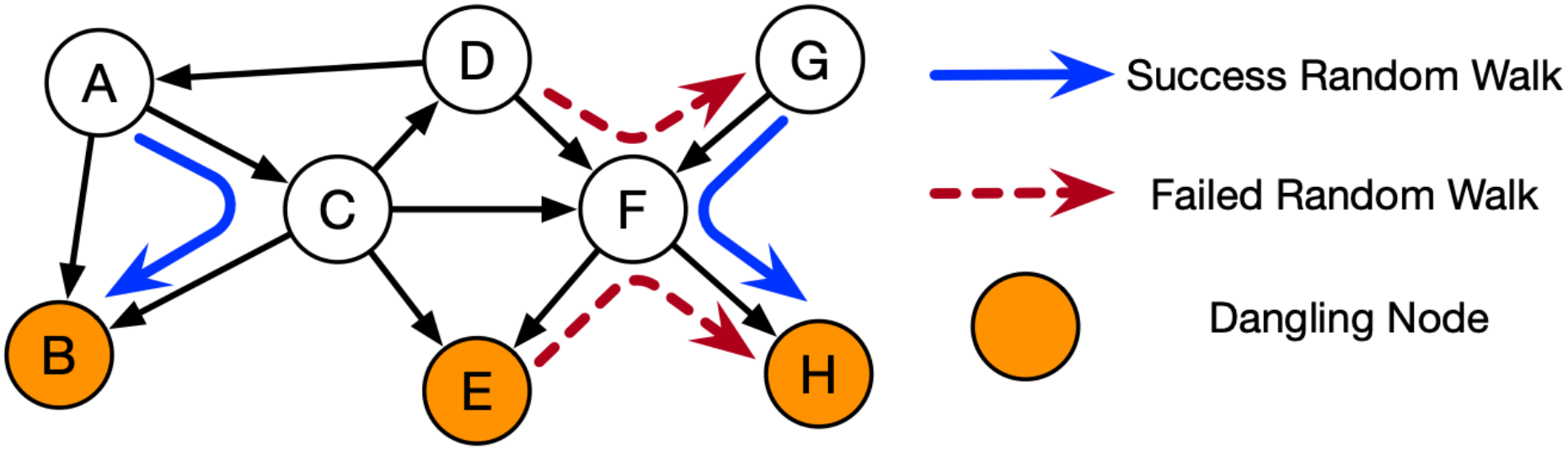
| DataSet | arxiv | | | | | | amazon | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P@10 | R@10 | P@20 | R@20 | P@50 | R@50 | P@10 | R@10 | P@20 | R@20 | P@50 | R@50 |
| CNbrs | 0.092 | 0.493 | 0.065 | 0.635 | 0.044 | 0.793 | 0.080 | 0.551 | 0.052 | 0.677 | 0.034 | 0.773 |
| Adar | 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** |

Table 6: Precision and Recall for top-k Node Recommendation, Directed Graph

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

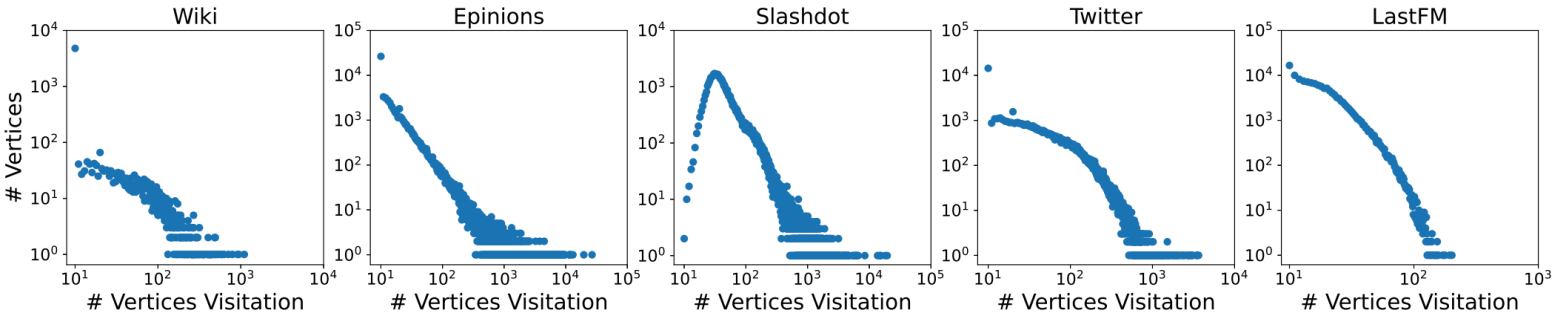



有向网络中的随机游走

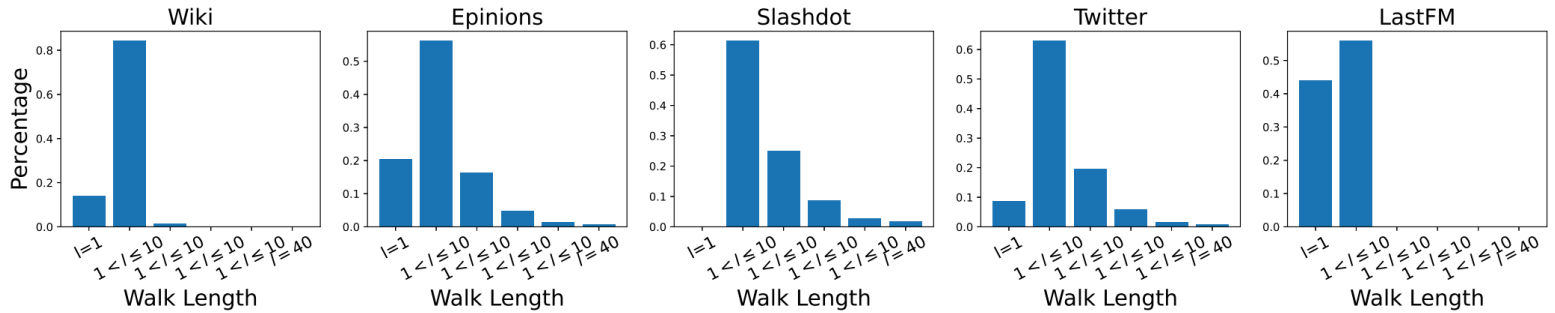
[3]Direction-Aware User Recommendation Based on Asymmetric Network Embedding(TOIS 2021)

# 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




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

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

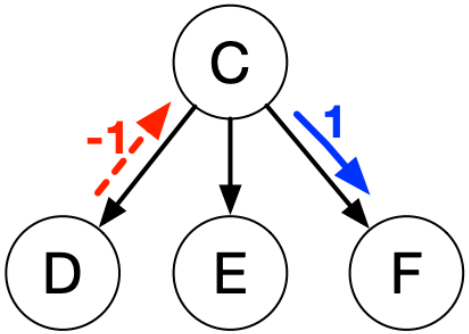
- 每一步随机游走时边的方向使用参数记录信息

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

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

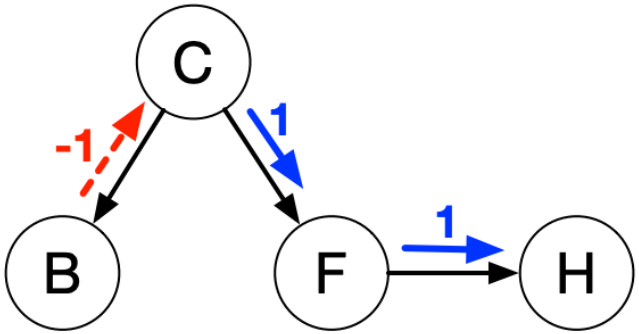
$$s_{i,i+k}=\frac{1}{k}\sum_{j=i}^{i+k-1} r_{j,j+1}$$

# InfoWalk 随机游走



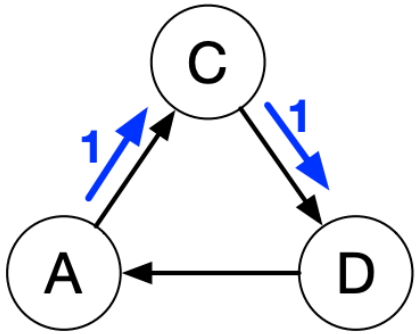

$S_{D\to C\to F}=0$

**(a)**



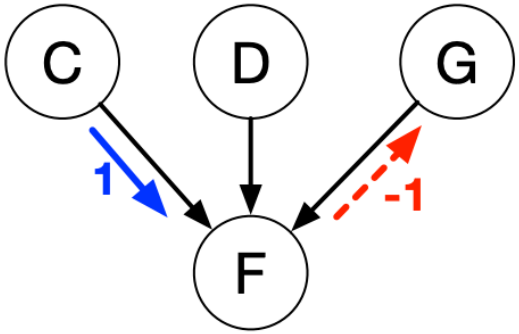

$S_{B\to C\to F\to H}=1$

**(c)**



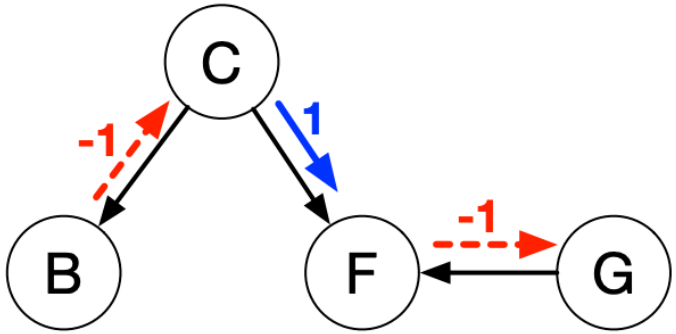

$S_{A\to C\to D}=2$

**(e)**



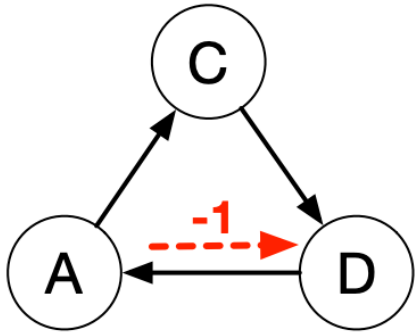

$S_{C\to F\to G}=0$

**(b)**




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

**(d)**

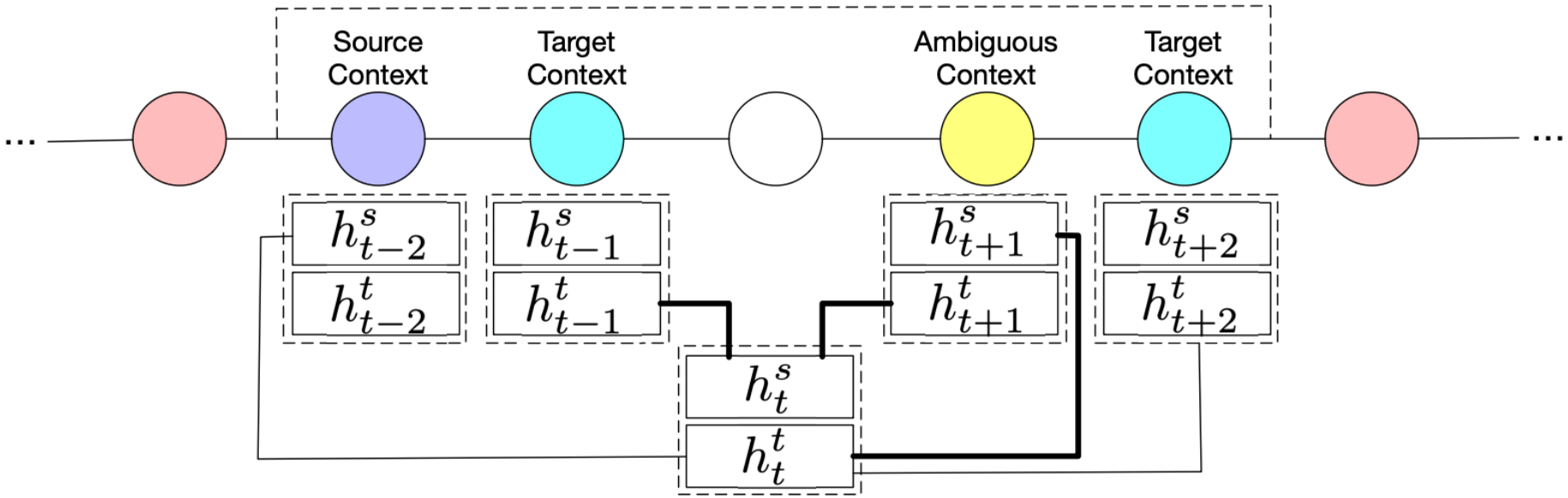



$S_{A\to D}=-1$

**(f)**

InfoWalk 随机游走

# 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

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

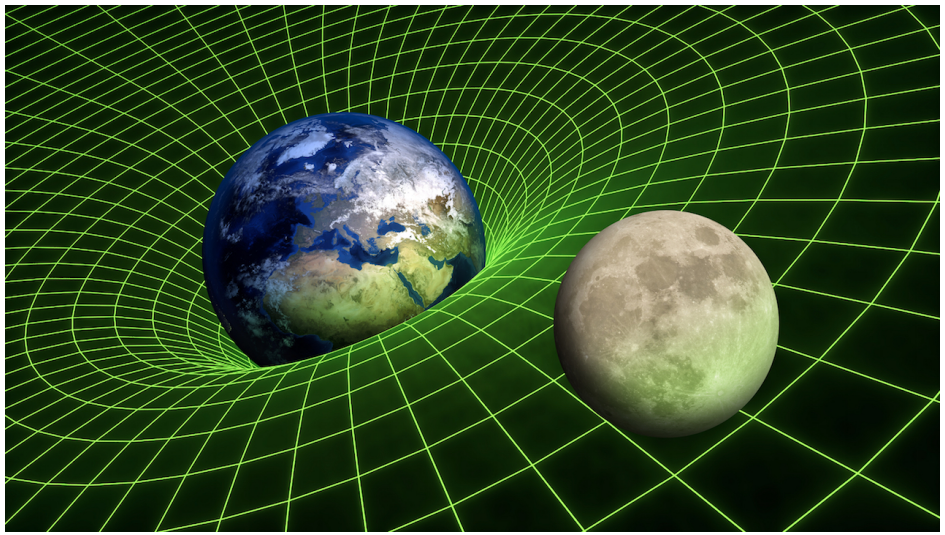
$$\max_{\mathrm{H}^s, \mathrm{H}^t} \sum_{u \in \mathrm{V}} \sum_{v \in DC_u} \log P\left(v \mid u, \pi_{u,v}\right) = \log \frac{\pi_{u,v} \cdot \exp\left(h_u^s \cdot h_v^t\right)}{\sum_{k \in V} \exp\left(h_u^s \cdot h_v^t\right)}$$

# 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

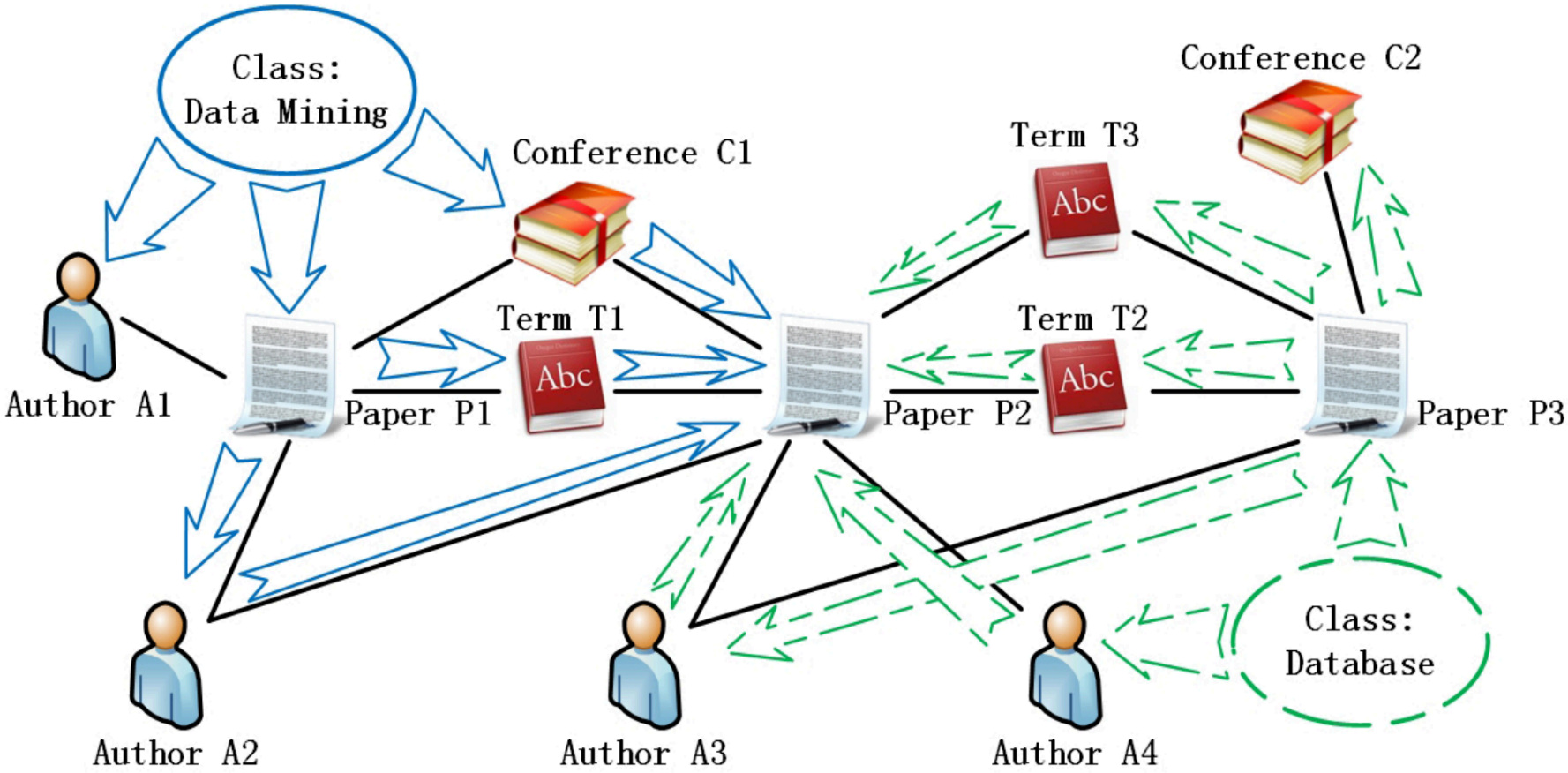
未来展望：

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?
2. 拉普拉斯矩阵的对称性在有向图中被破坏，如何使用图卷积操作?

$$\mathcal{L}_{pr} = \mathbf{I} - \frac{1}{2}\left(\mathbf{\Pi}_{pr}^{\frac{1}{2}}\mathbf{P}_{pr}\mathbf{\Pi}_{pr}^{-\frac{1}{2}} + \mathbf{\Pi}_{pr}^{-\frac{1}{2}}\mathbf{P}_{pr}^{T}\mathbf{\Pi}_{pr}^{\frac{1}{2}}\right)$$

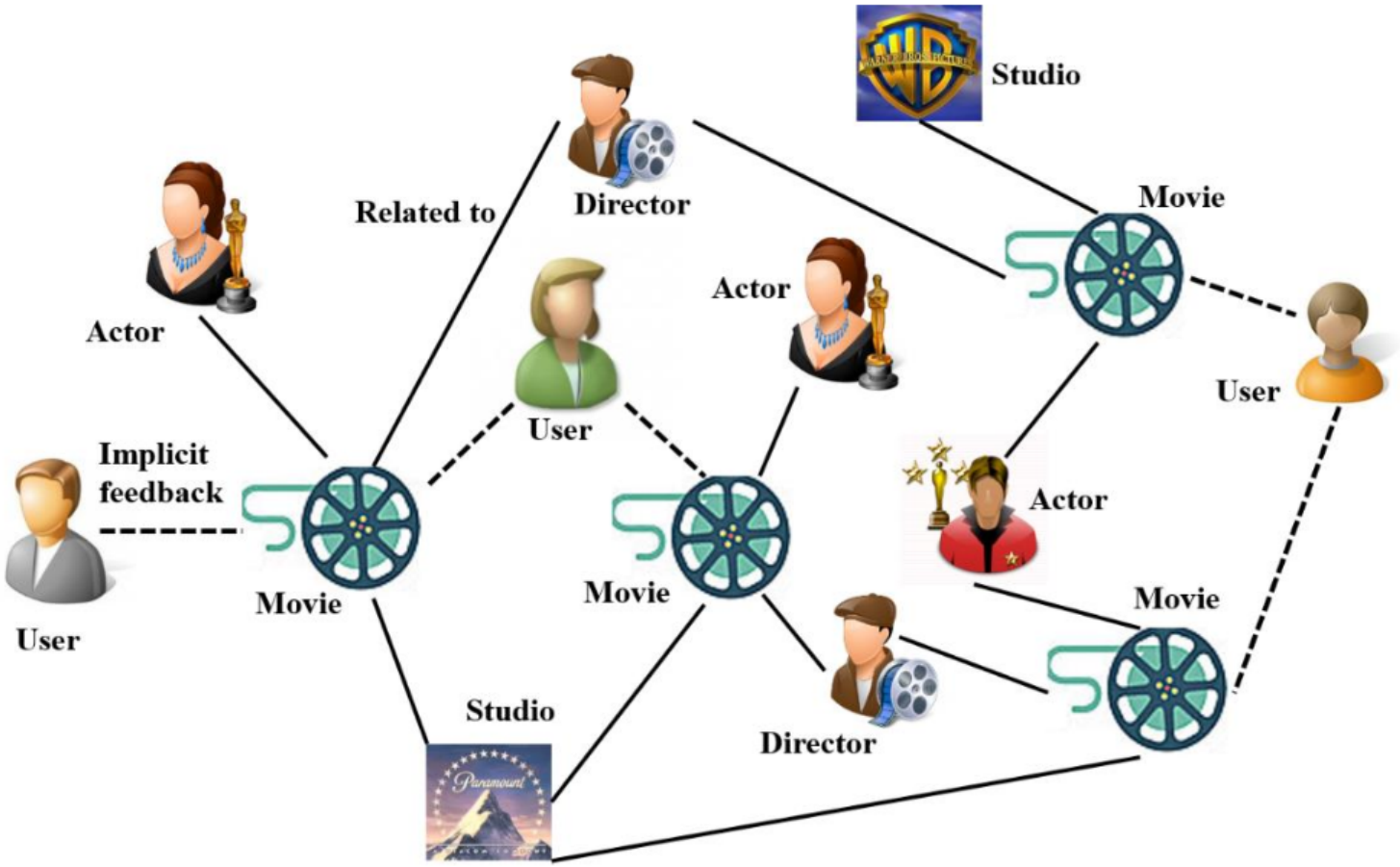
3. 如何使用一套表征刻画非对称关系?

# 异构信息网络




经典异构信息网络

# 异构信息网络

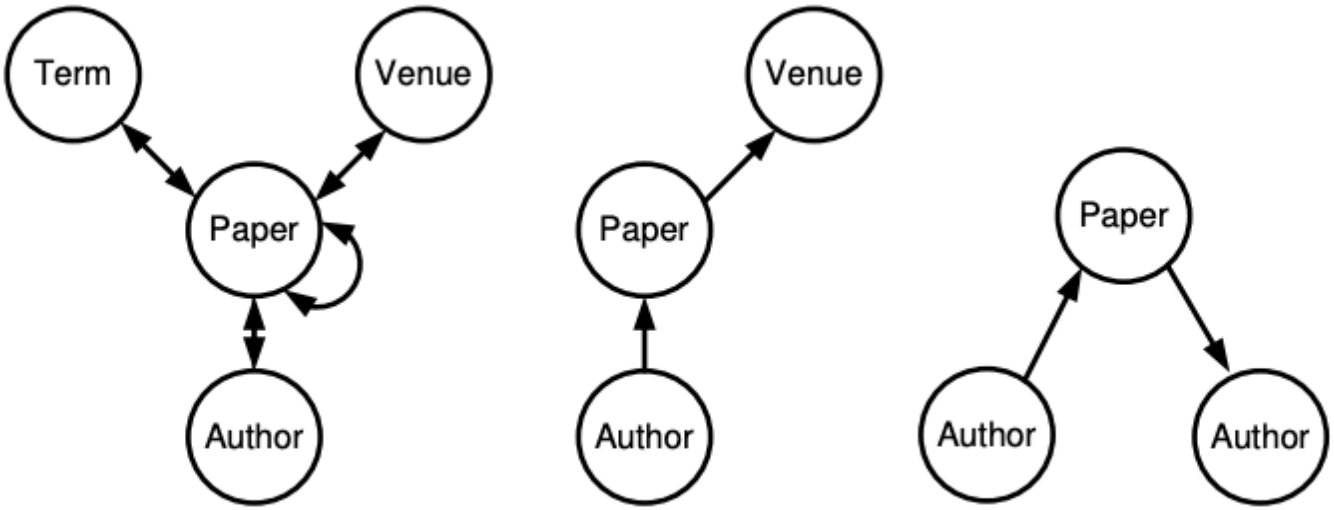



经典异构信息网络

# 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

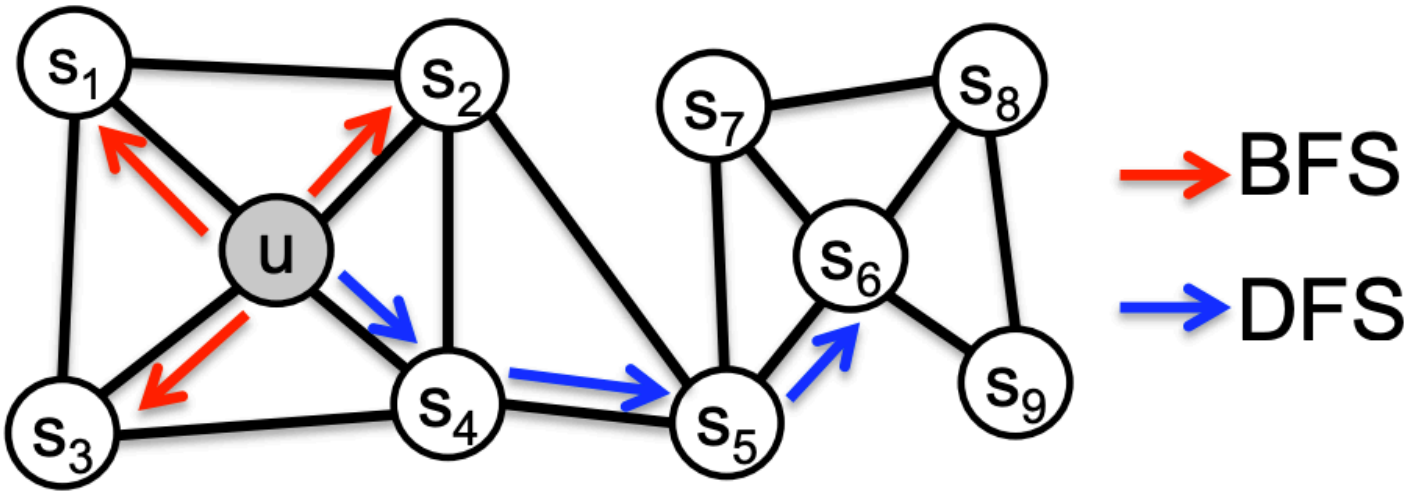
异构图神经网络的难点：

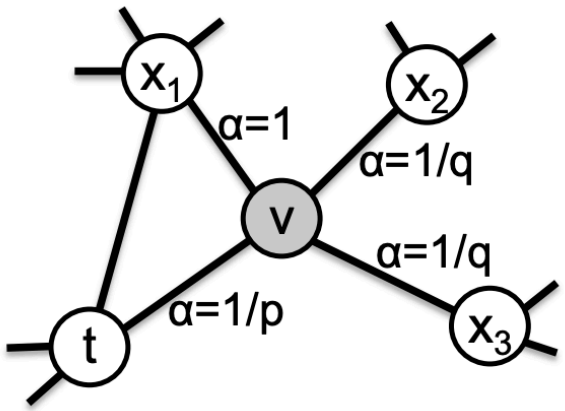
1. 如何定义和选择元路径?
2. 如何在元路径对应的语义空间中学习节点的表征?
3. 如何将不同元路径的表征融合?

发展历程：

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

# Node2Vec



# Metapath2Vec

## 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

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

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

# Metapath2Vec

## 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。

# Metapath2Vec

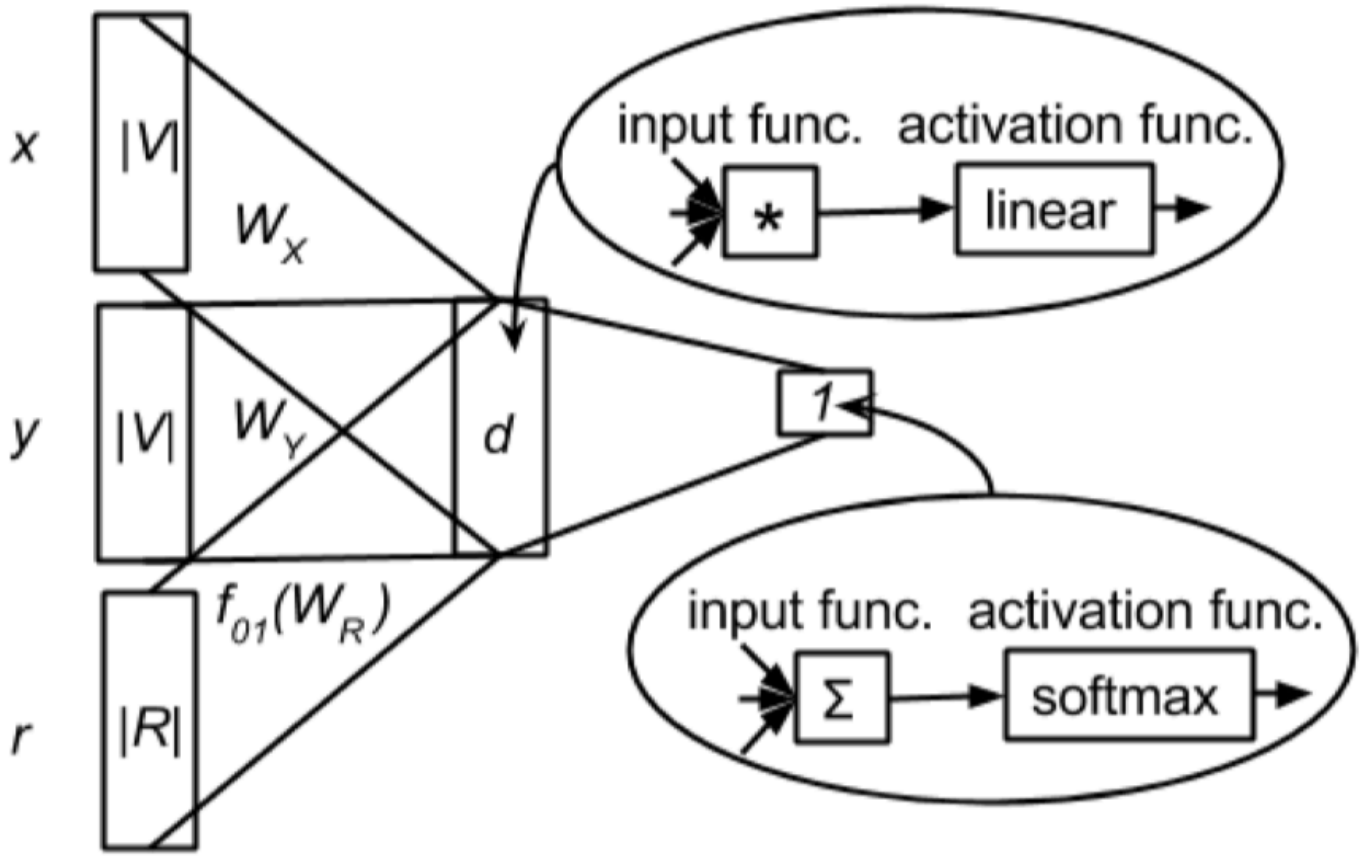
**Table 2: Multi-class venue node classification results in AMiner data.**

| 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 |
| | PTE | 0.8986 | 0.9023 | 0.9051 | 0.9061 | 0.9066 | 0.9068 | 0.9075 | 0.9077 | 0.9079 | 0.9082 |
| | *metapath2vec* | 0.9279 | 0.9319 | 0.9346 | 0.9356 | 0.9361 | 0.9365 | 0.9365 | 0.9365 | 0.9367 | 0.9369 |
| | *metapath2vec++* | 0.9173 | 0.9217 | 0.9243 | 0.9254 | 0.9259 | 0.9261 | 0.9261 | 0.9262 | 0.9264 | 0.9266 |

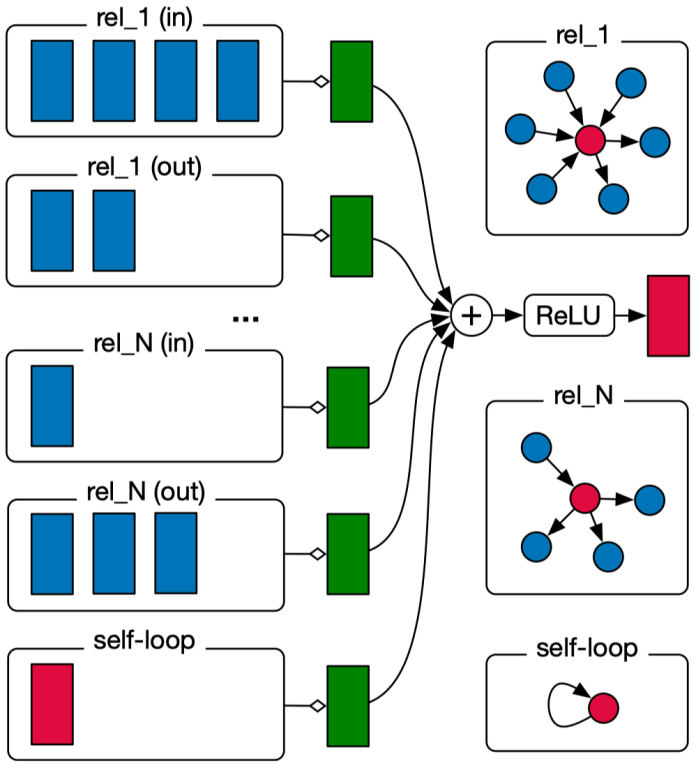
# HIN2Vec




HIN2Vec[5]

[5]HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning(CIKM2017)

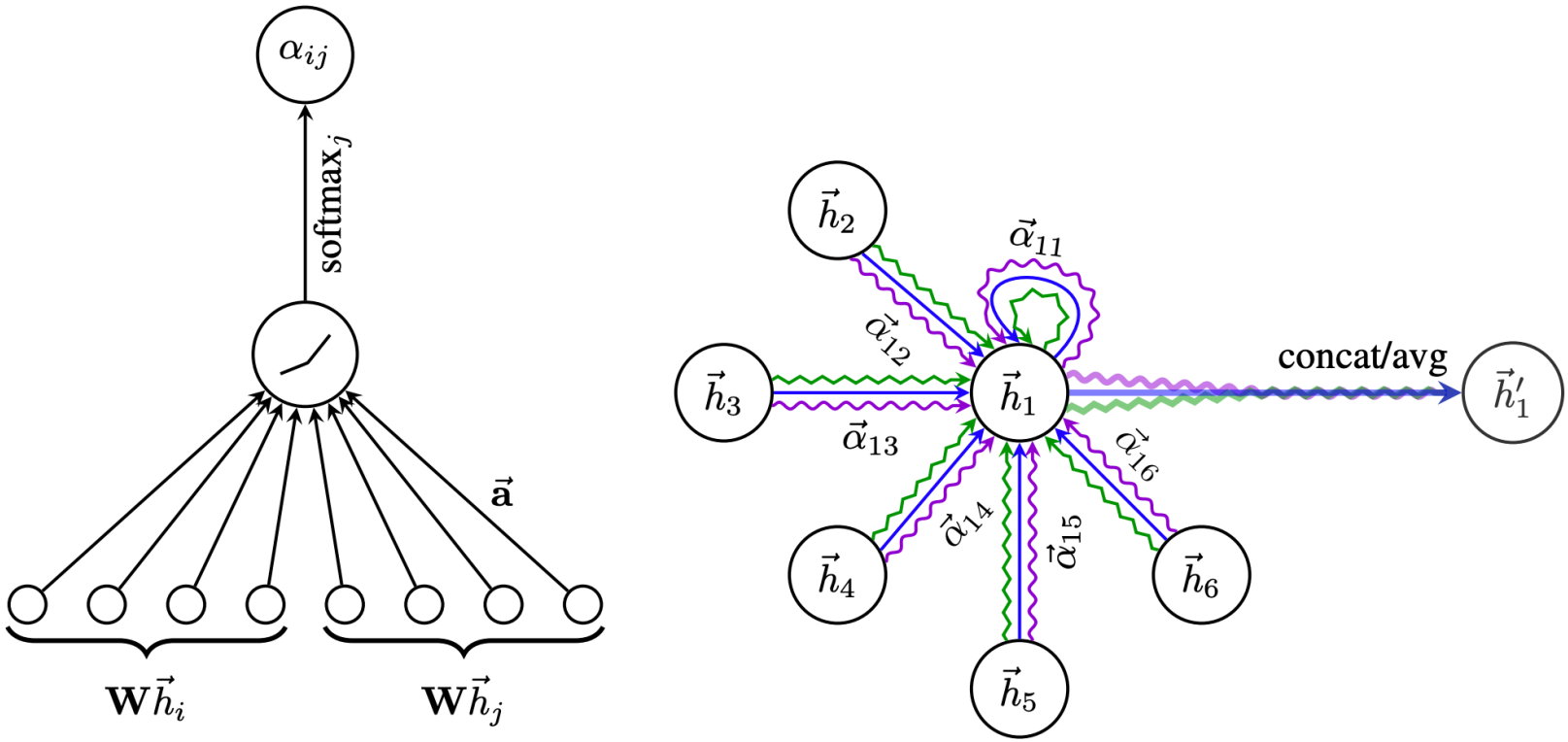
# RGCN[6]




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

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

# Graph Attention Network




Graph Attention Network

如何在异构网络上整合邻居信息?

# Heterogenous Graph Attention Network

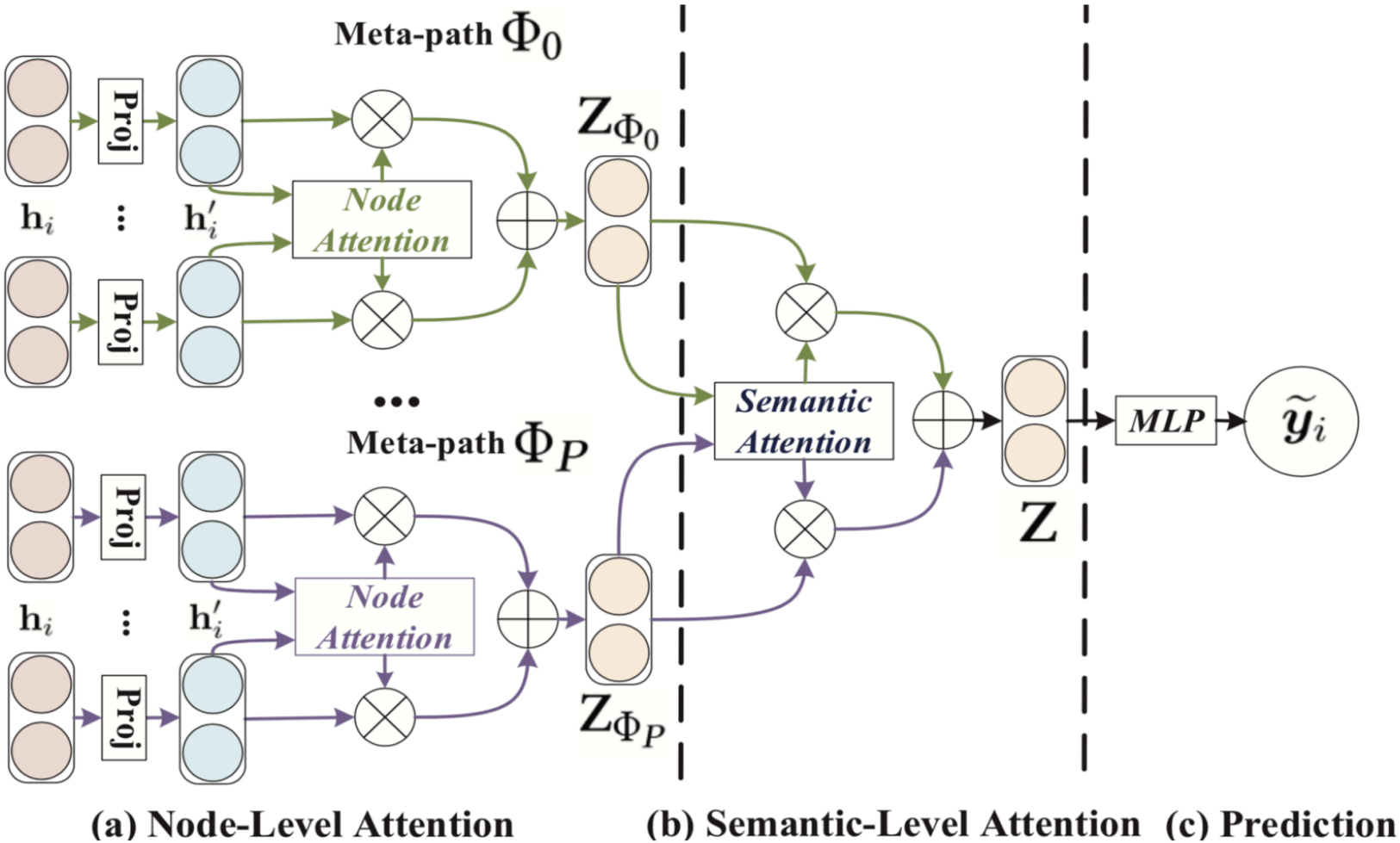
## 异构图神经网络的特点

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

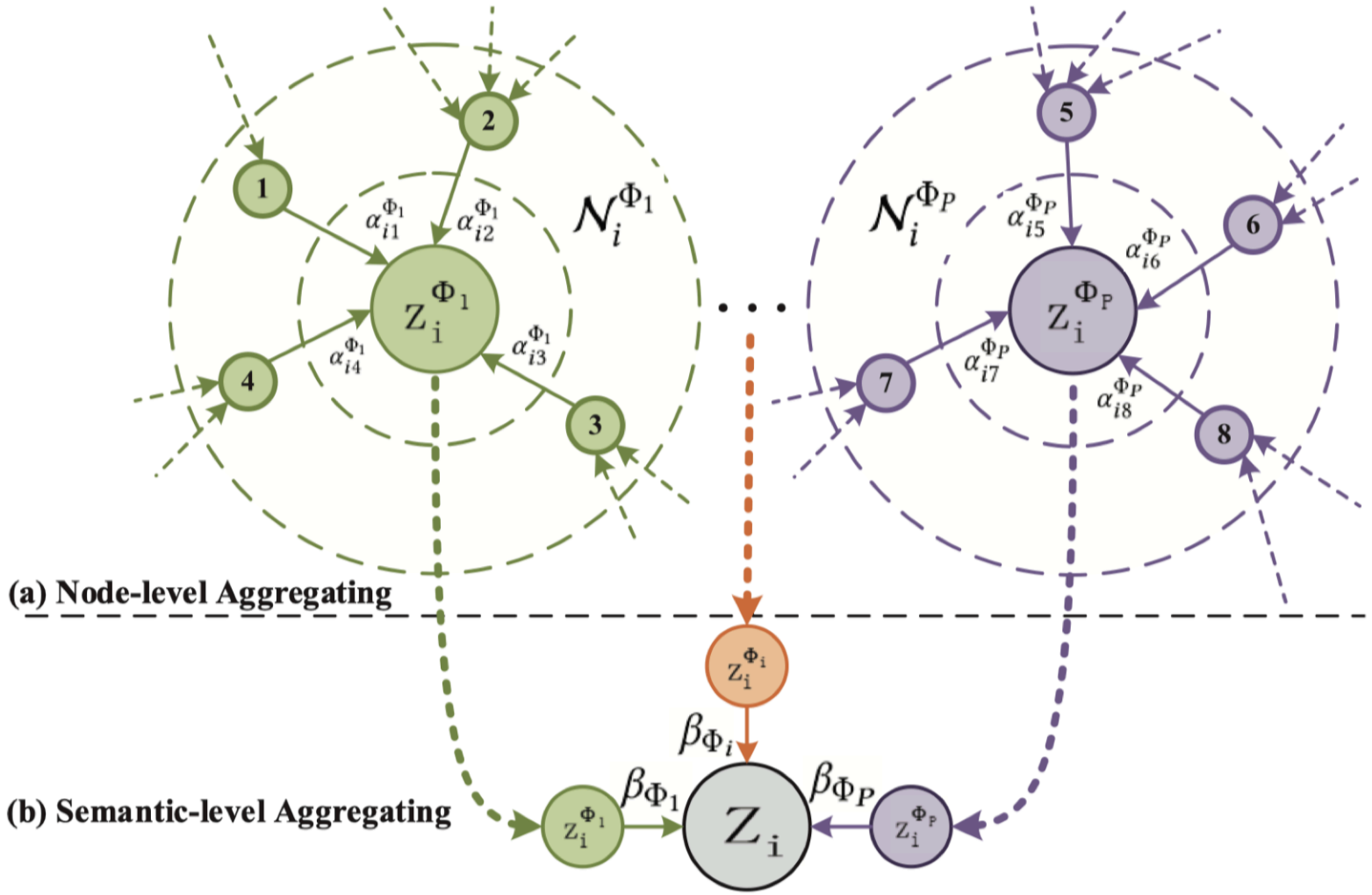
1. 使用 meta-path 异构信息网络投影到多个同构图
2. 在每个同构图内使用注意力网络整合邻居信息
3. 对不同 meta-path 对应的多个同构图使用全局注意力机制

# Heterogenous Graph Attention Network




HAN 模型架构

# 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

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

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

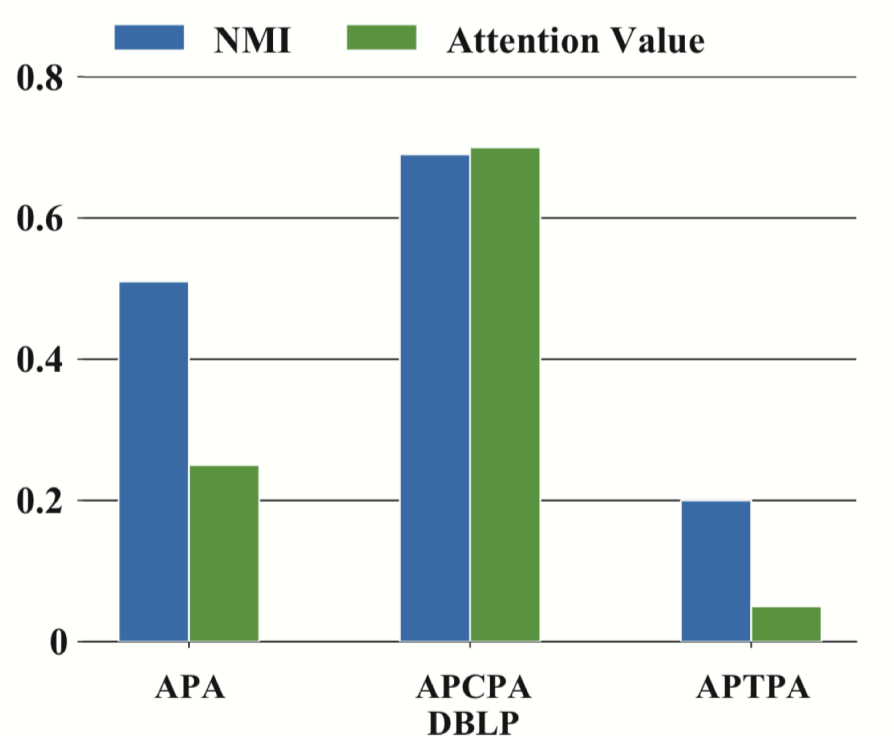
$$\mathbf{Z} = \sum^{P} \beta_{\Phi_p} \cdot \mathbf{Z}_{\Phi_p}$$

# Heterogenous Graph Attention Network

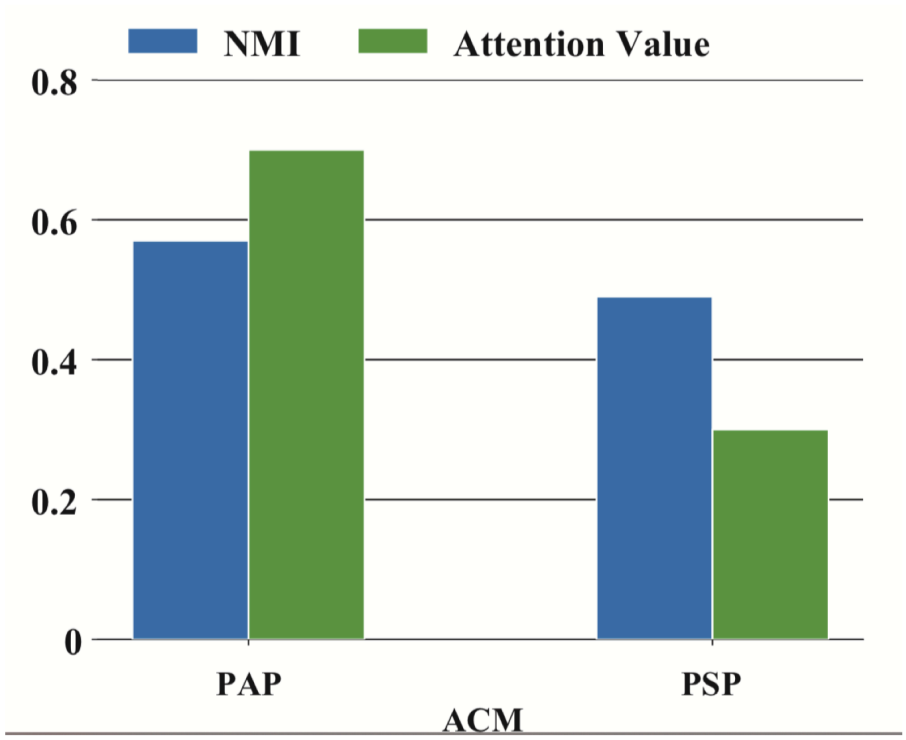
**Table 3: Qantitative results (%) on the node classification task.**

| 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



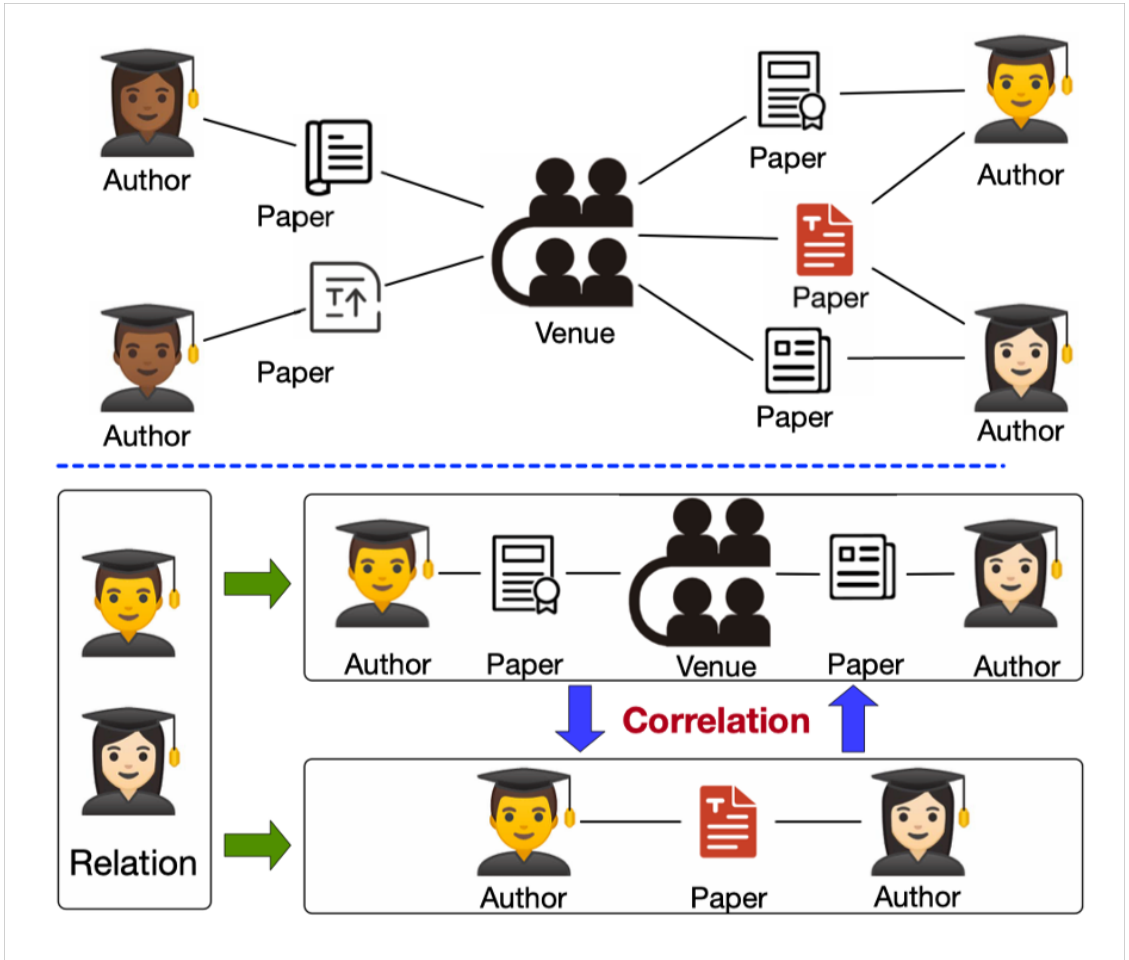

(a) NMI values on DBLP




(b) NMI values on ACM

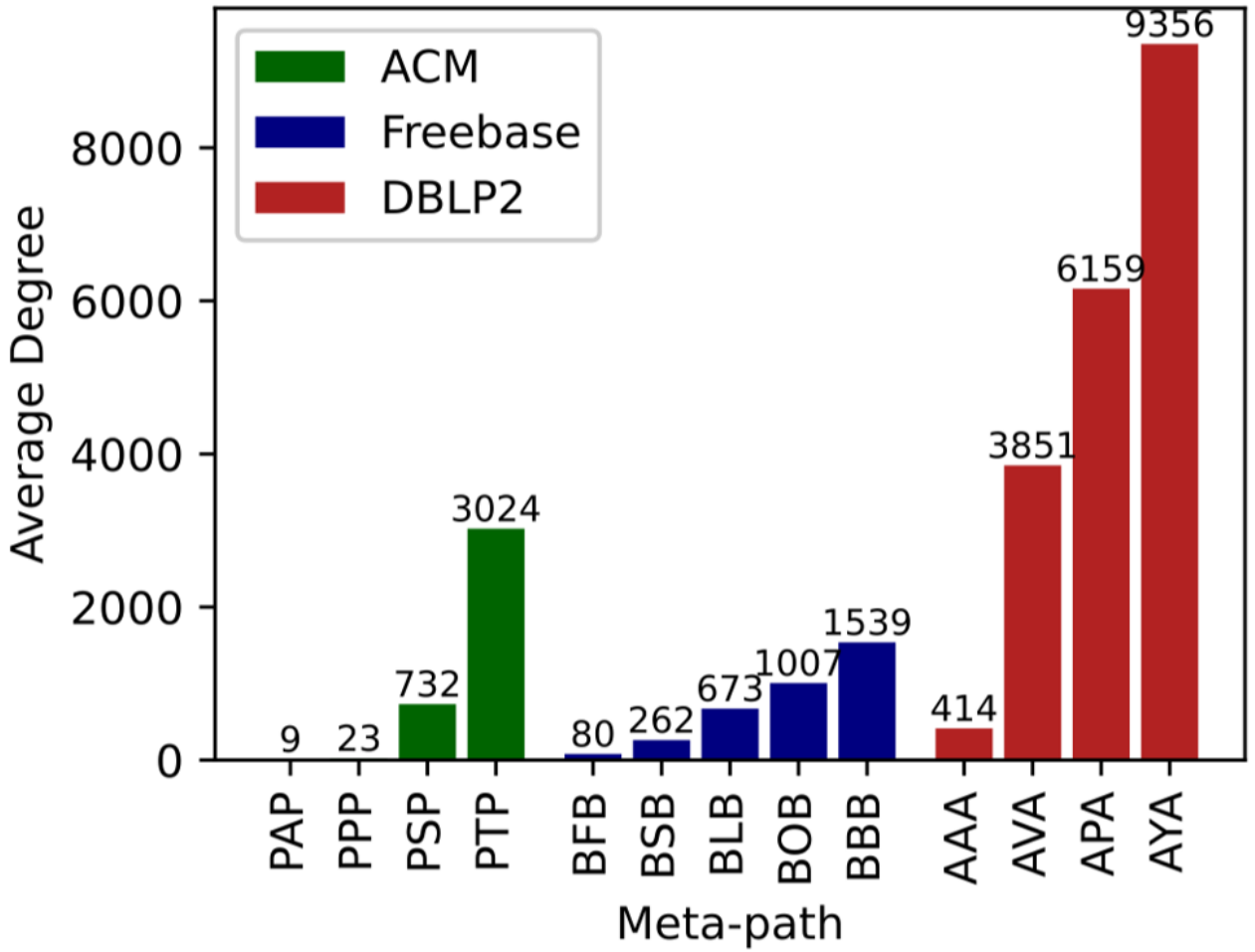
Attention 权重与 meta-path 质量对比

# Collaborative Knowledge Distillation for Heterogeneous Information Network Embedding



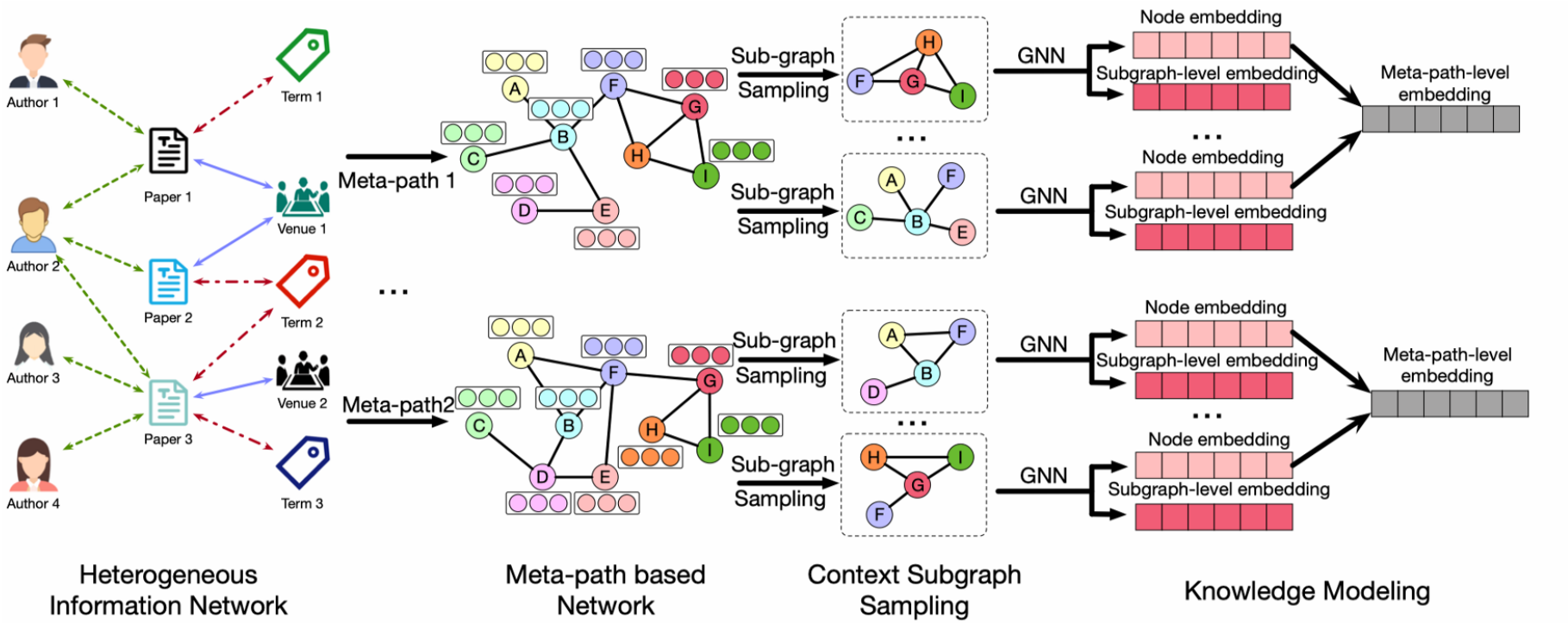

Meta-path 之间的关联

# CKD



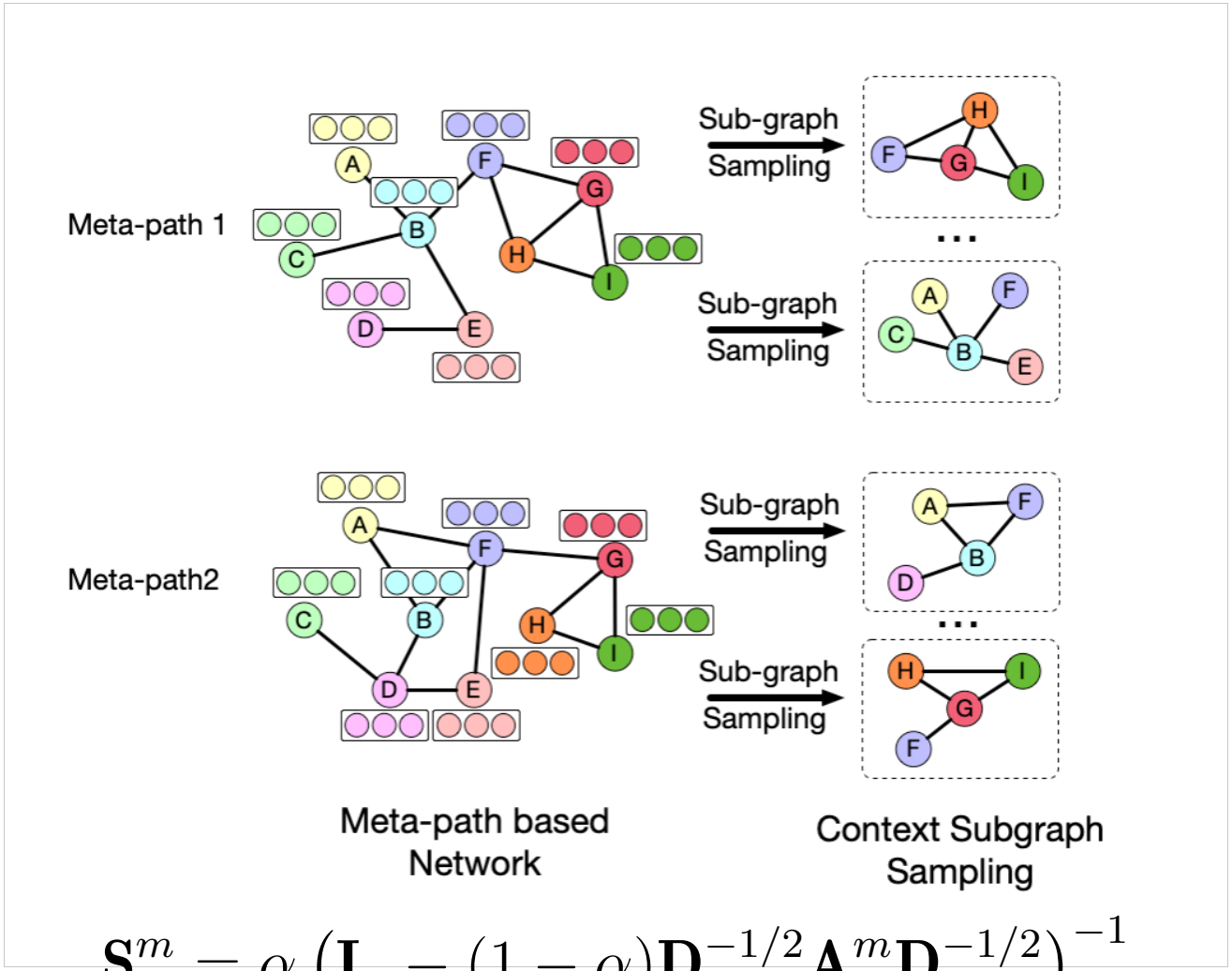

Meta-path 之间的差异：Sparsity 和 Redundancy

# 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:

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

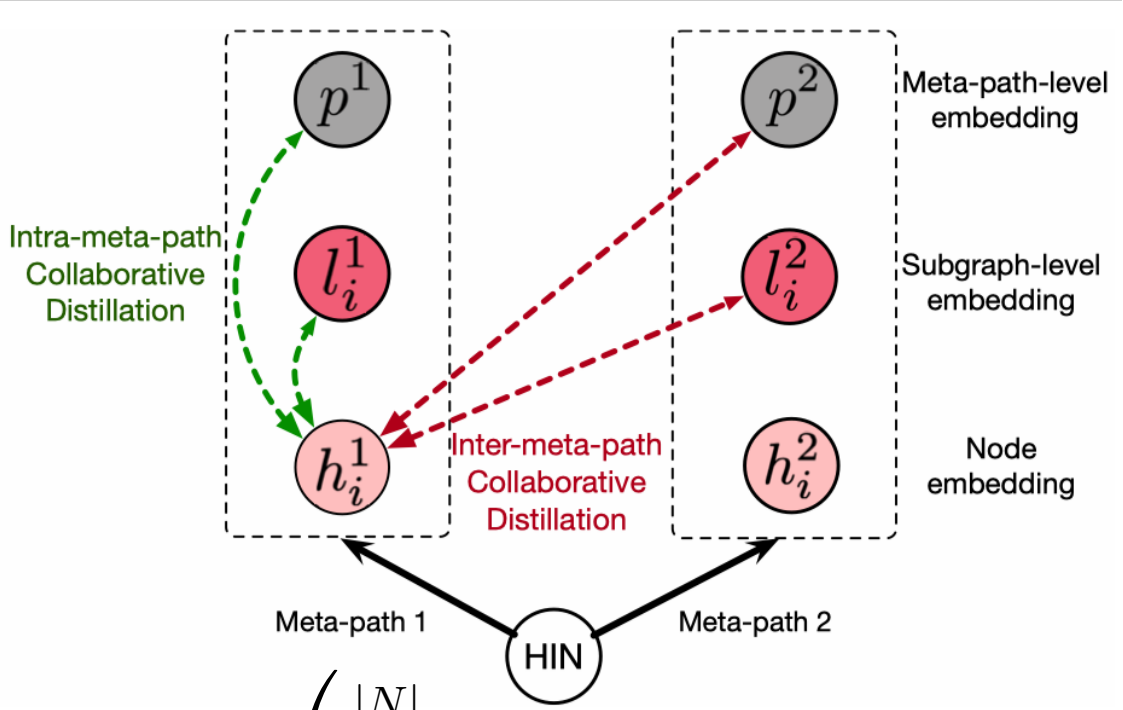
Subgraph-level Knowledge:

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

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 | - | - | - | - |
| HetGNN | 85.7±0.1 | 85.6±0.1 | 92.0±0.6 | 92.3±0.7 | - | - | - | - | - | - | - | - |
| | 86.1±0.1 | 86.1±0.1 | 92.3±0.5 | 92.6±0.5 | - | - | - | - | - | - | - | - |
| | 86.6±0.2 | 86.7±0.2 | 92.8±0.6 | 93.1±0.5 | - | - | - | - | - | - | - | - |
| CKD | **91.9**±0.4 | **91.9**±0.4 | 92.5±0.2 | 92.8±0.2 | **69.7**±0.5 | **79.7**±0.8 | **36.8**±1.1 | **39.3**±1.6 | 48.2±0.7 | 60.5±0.4 | **90.2**±0.3 | **90.1**±0.3 |
| | **92.9**±0.3 | **92.9**±0.3 | 92.5±0.4 | 92.8±0.4 | **65.6**±0.3 | **77.9**±0.1 | **37.4**±0.9 | **40.1**±0.6 | **49.6**±0.4 | **61.1**±0.7 | **90.4**±0.3 | **90.3**±0.3 |
| | **92.8**±0.8 | **92.7**±1.0 | 92.3±0.4 | 92.6±0.4 | **70.4**±0.5 | **80.2**±0.6 | 37.8±1.2 | 40.4±1.2 | 48.1±0.8 | 60.4±0.5 | **90.2**±0.2 | **90.1**±0.1 |

# 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

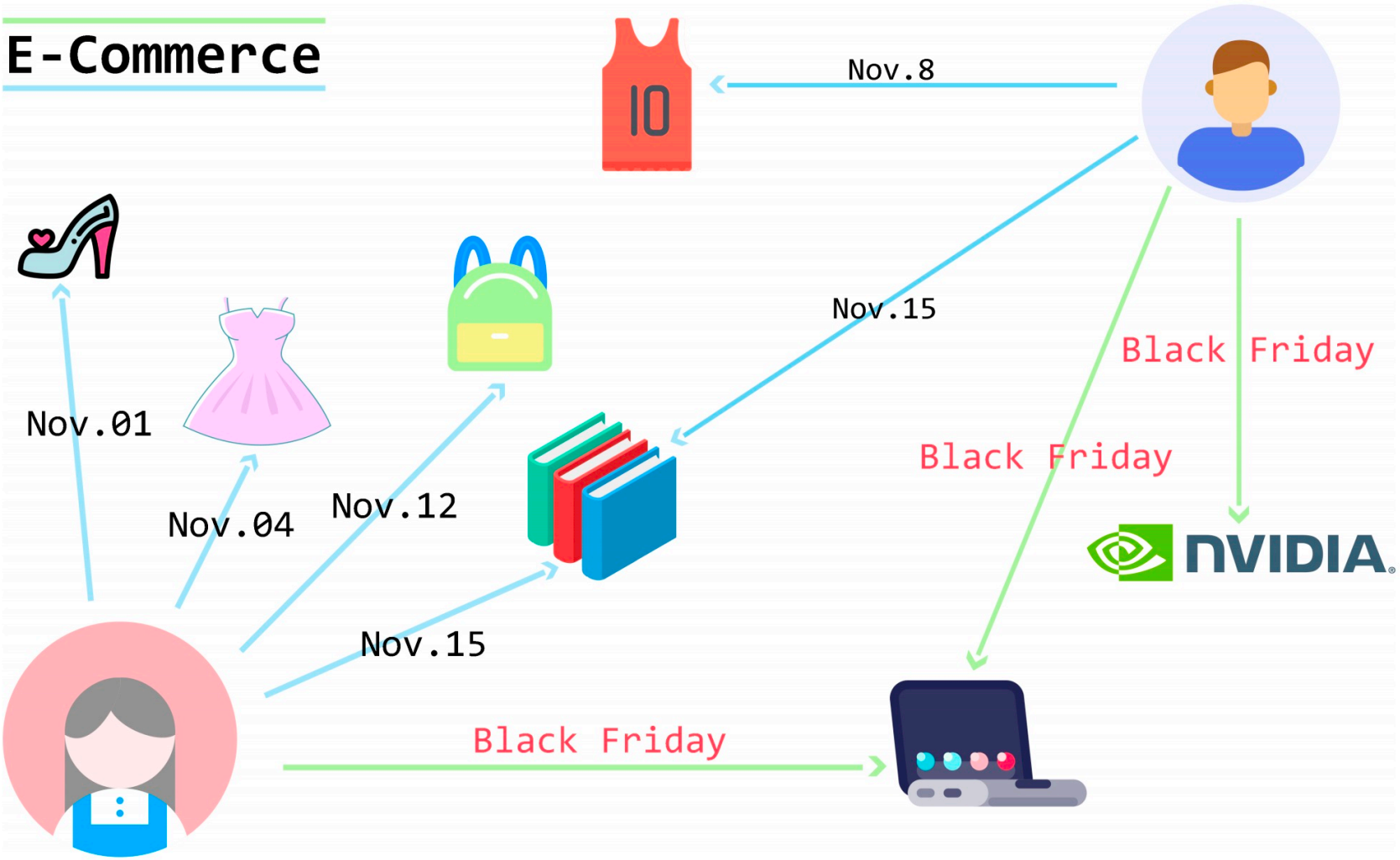
## 动态网络

- 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。

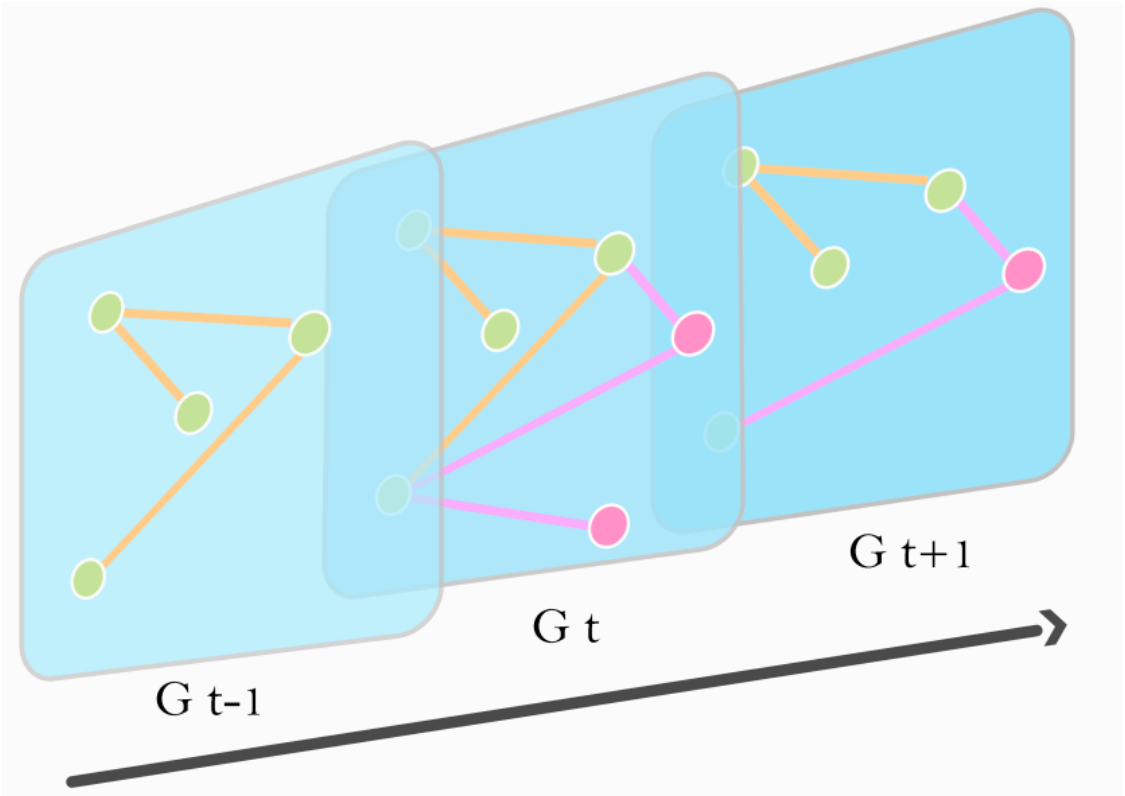
## 高效的表示更新方法

- 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

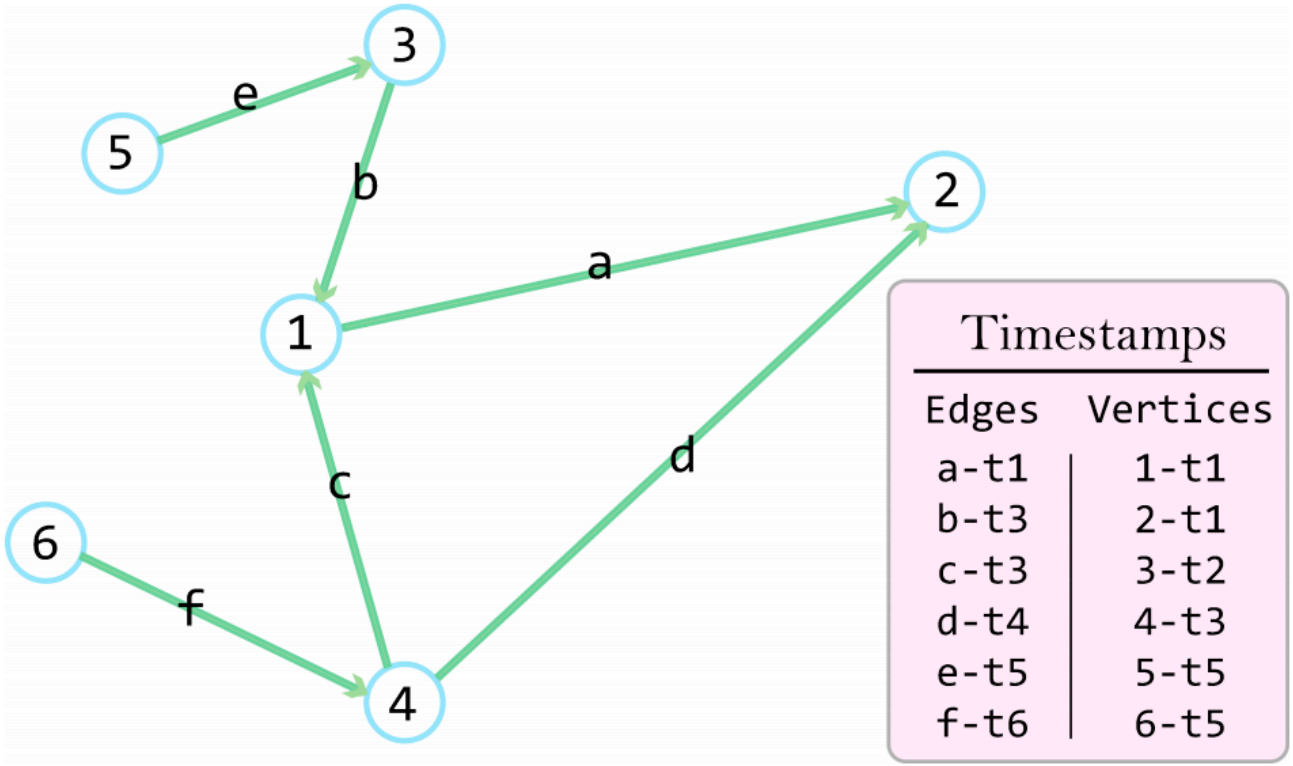
## 离散模型

- 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

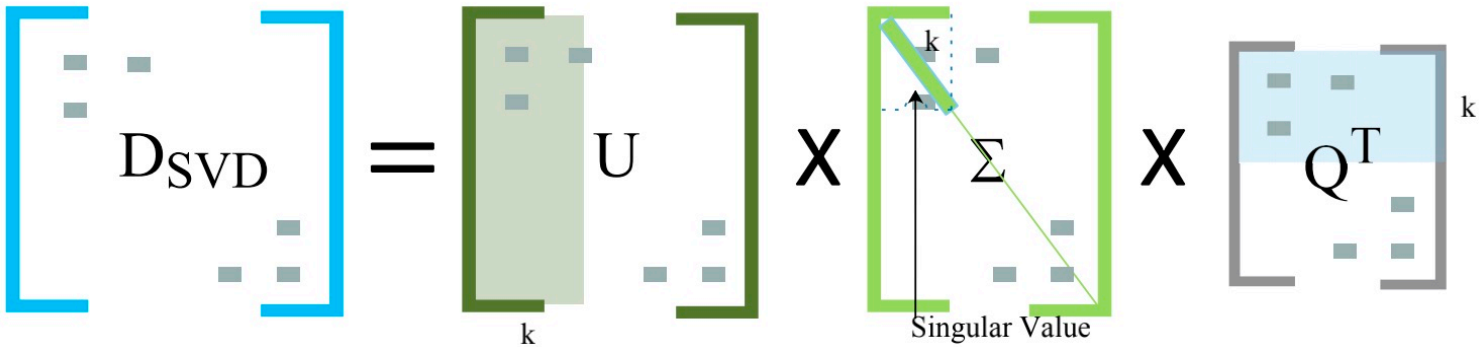
矩阵分解是在网络嵌入领域最普遍的降维方法。其中最具代表性的是奇异值分解（Singular value decomposition，SVD）。

## SVD

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

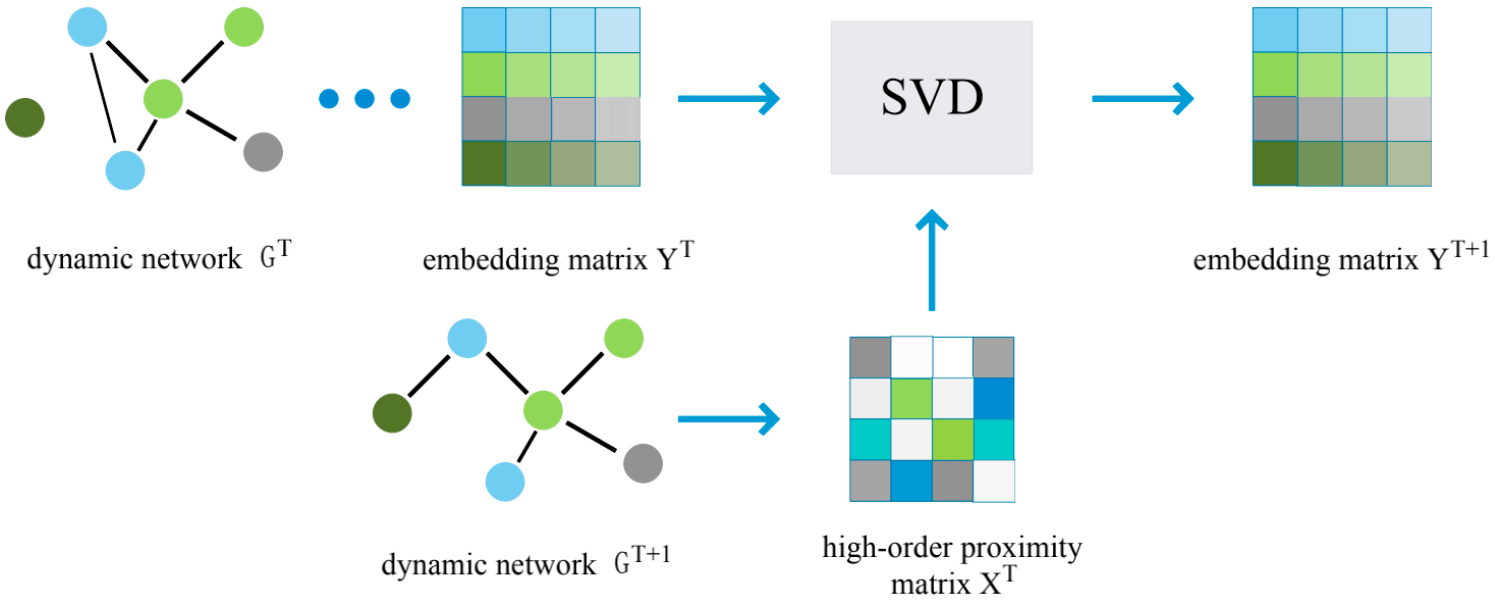
$$D_{\mathcal{SVD}} = U \Sigma Q^T$$

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

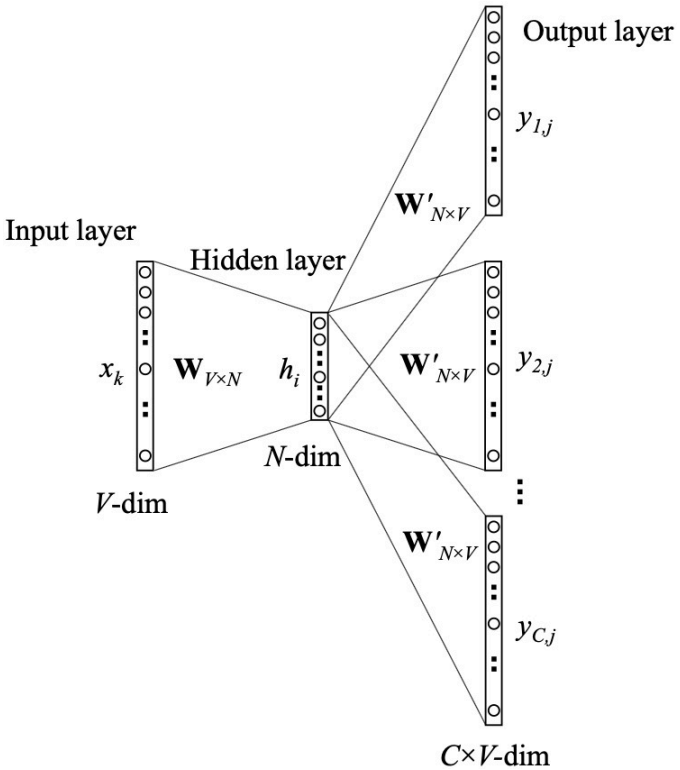
从矩阵分解的角度看，网络的动态演化相当于被分解矩阵的不断变化，因此可以根据矩阵扰动理论更新。



# 基于 Skip-Gram 的嵌入方法

## Skip-Gram

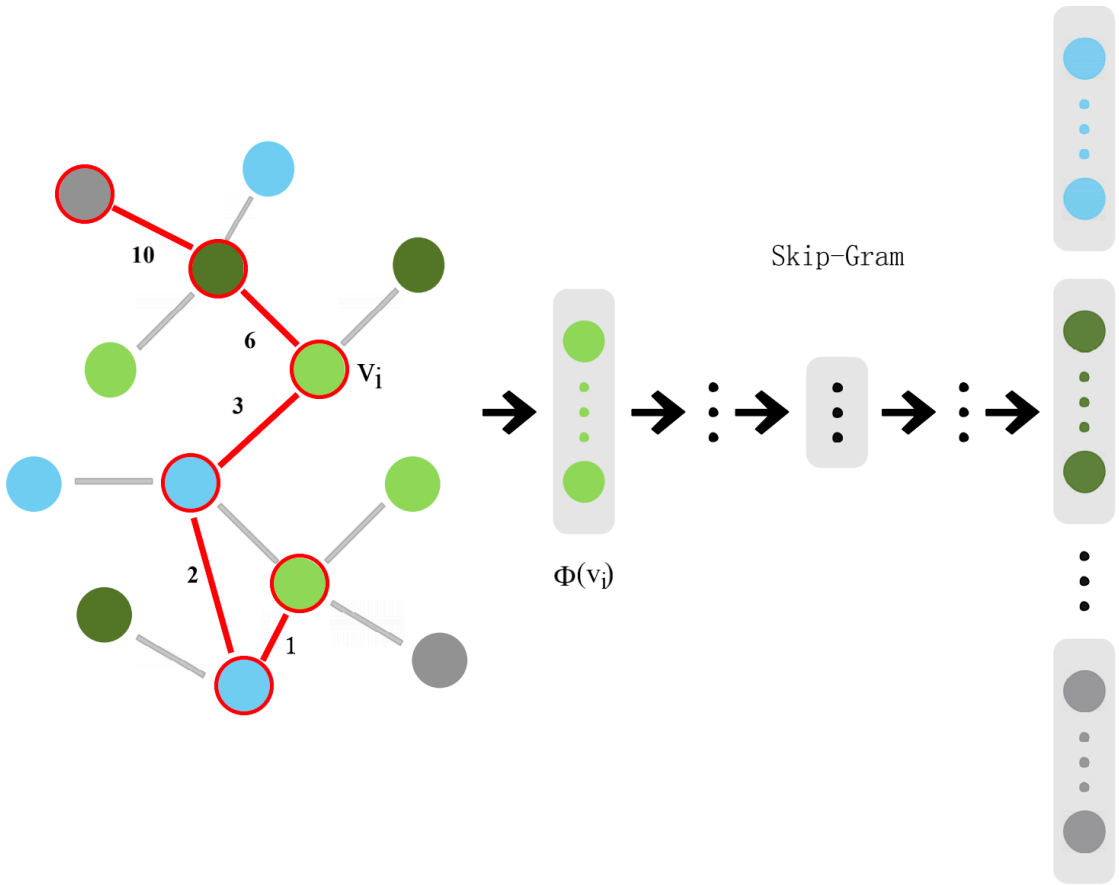
- Word2Vec 的一种经典模型，可以通过输入词来预测上下文。



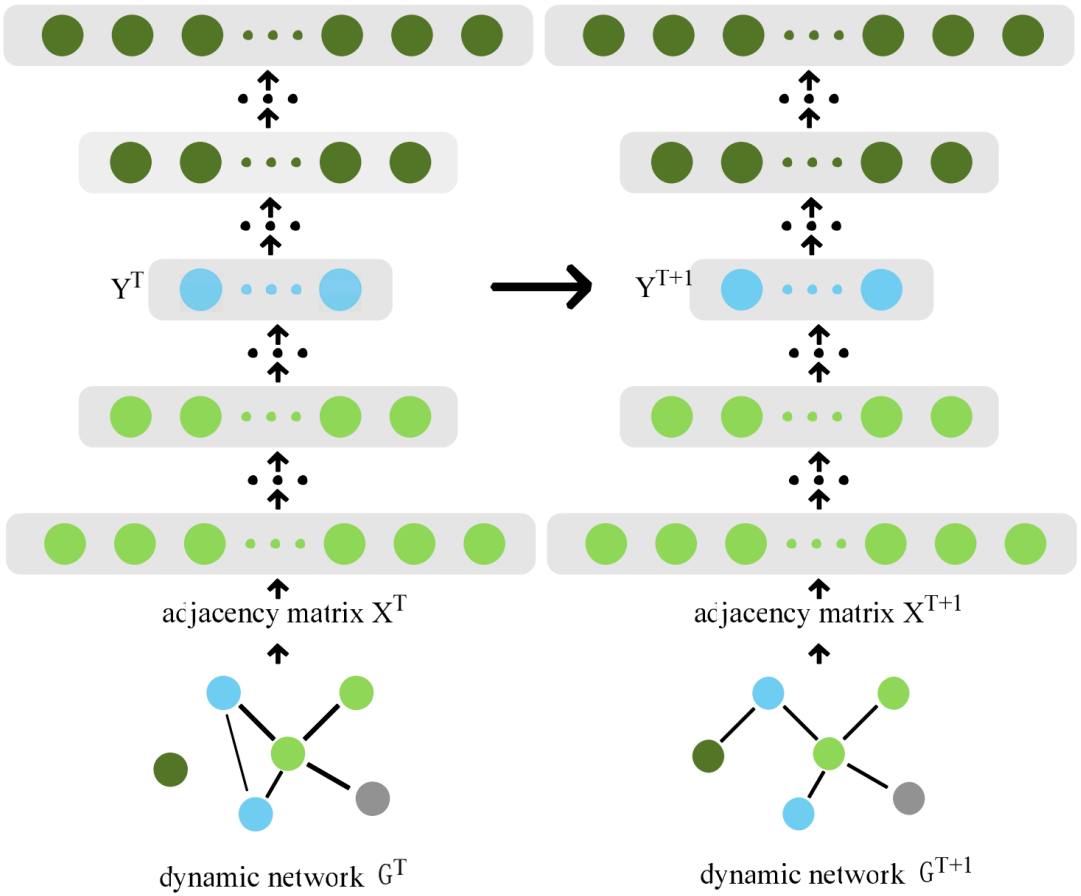

- DeepWalk 提出可以将节点看作为单词，游走序列则对应着句子。
- 由此，大量基于该模型的静态网络嵌入方法被提出。

# 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

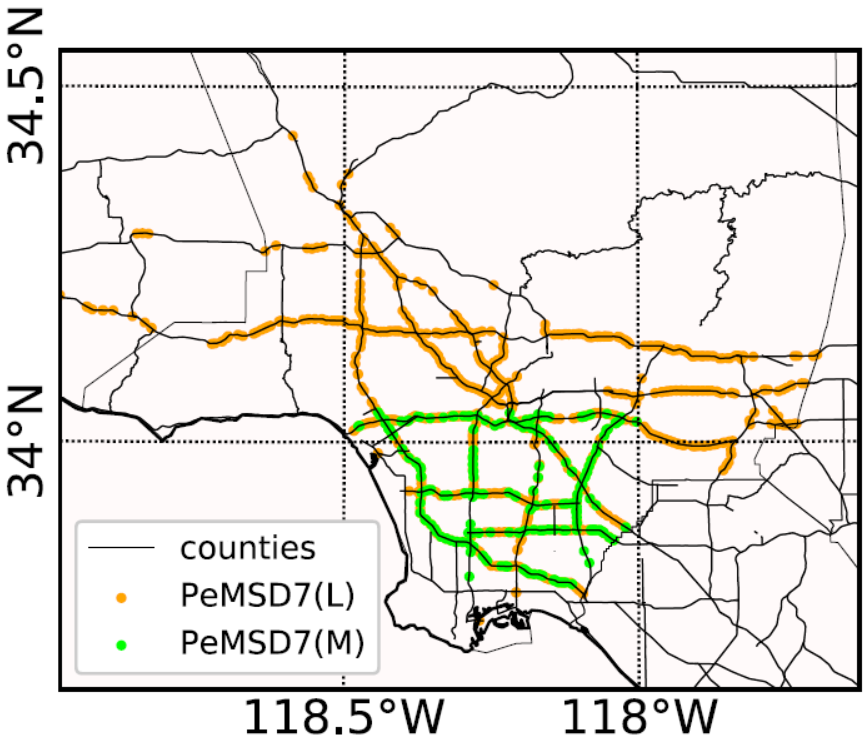



- 离散模型
- 将上一时刻 AE 的权重参数作为下一时刻网络的初始化

# STGCN-动态交通流量图预测

## 交通流量预测

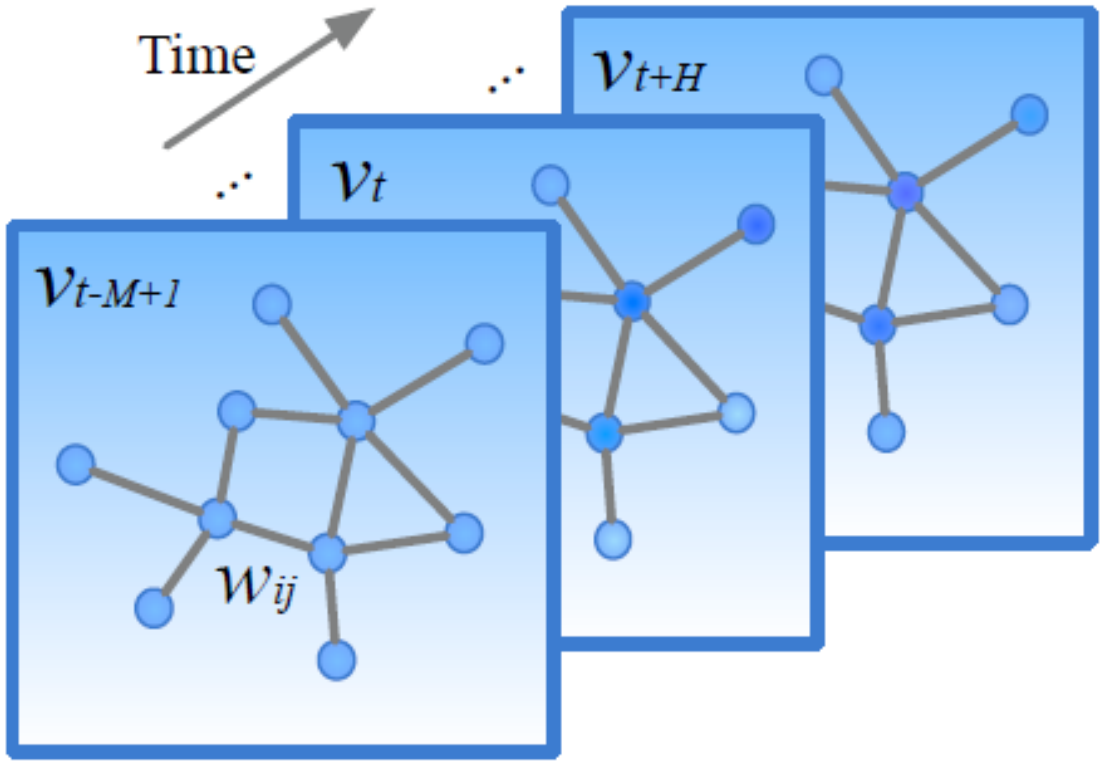
把路口的车流量看作图中节点的属性，道路看作连接节点的边，不同时刻的交通流量网就构成了经典的离散动态图。



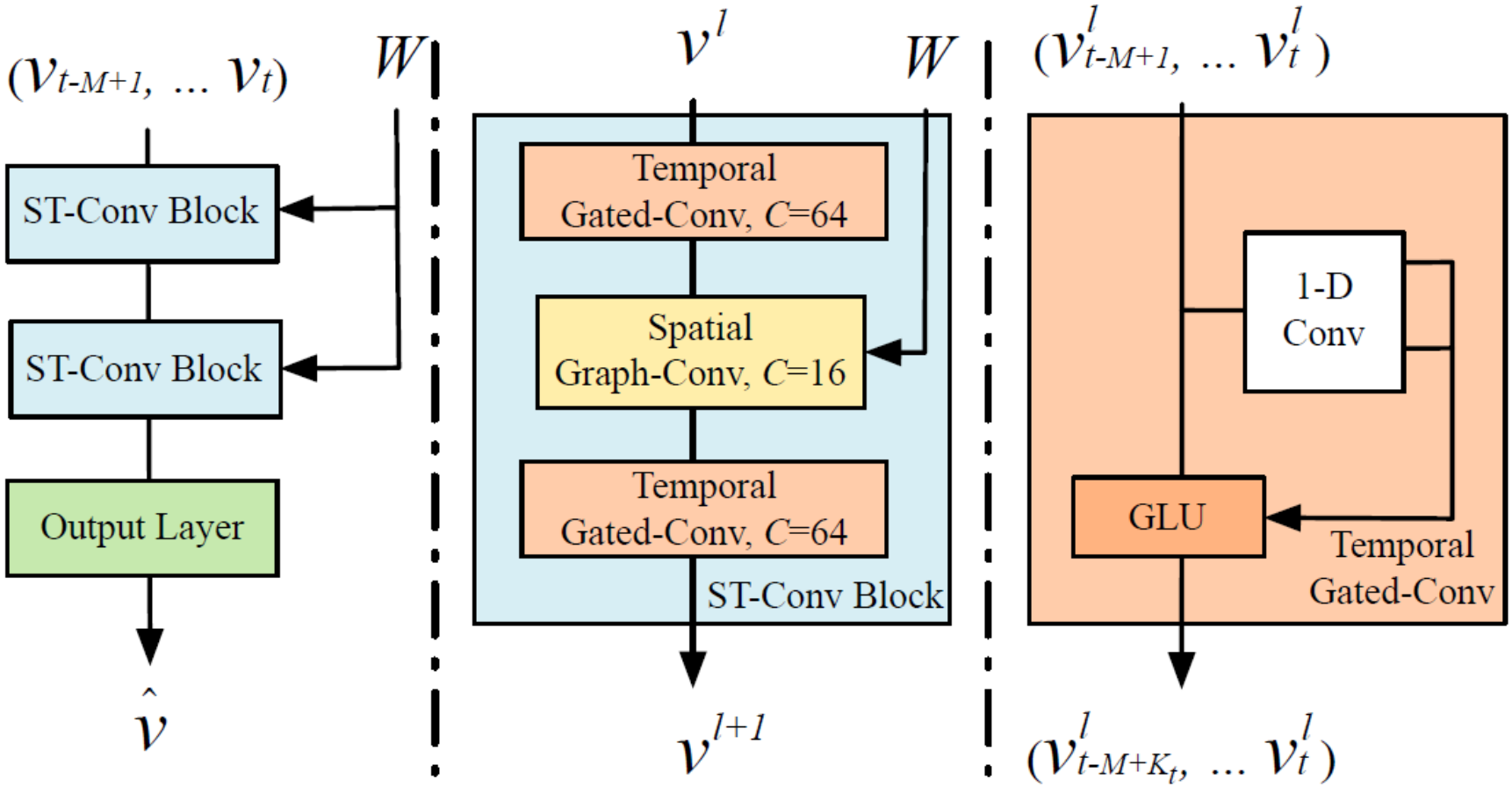
# STGCN-动态交通流量图预测

## 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 $Y \in R^{(M-K_t+1) \times n \times C_o}$ 。

# STGCN-动态交通流量图预测

## 空域卷积块

- 空域卷积在每个时间步的图上进行 (不在时间步之间进行)。
  输入

$$X \in R^{n \times C_i}$$

  按照 ChebGCN，输出

$$Y = \sum_{i=0}^{K-1} \theta_i T_i(\tilde{L}) X$$

- 其中 $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$ $\tilde{L} = \frac{2L}{\lambda_{\max}} - I_n$ ，$L = I_n - D^{-1/2} A D^{-1/2}$ ，$A$ 是邻接矩阵，论文中 $K = 3$ 。卷积核 $\Theta \in R^{K \times C_i}$ ，个数为 $C_o$ 。对于一张完整的时空图:
- 输入 $X \in R^{M \times n \times C_i}$ ，输出 $Y \in R^{M \times n \times C_o}$ 。

# STGCN-动态交通流量图预测

- 经过了时域卷积块和空域卷积块后，原本的节点属性维度 $\mathcal{V}$ 已经不再是简单的流量特征，而是通过卷积聚合了时空信息的全新特征。

## 时空卷积块的输出层

- 根据时域卷积块的一维卷积，每经过一个时空卷积块，数据在时间维度的长度减小 $2\left(K_t-1\right)$ 。所以经过两个时空卷积块后，得到 $Y \in R^{(M-4(K_t-1))\times n\times C_o}$ 。
- 输出层包括一个时域卷积层和一个全连接层，时域卷积层的卷积核大小 $\Gamma \in R^{(M-4(K_t-1))\times C_o}$ ，个数为 $C_o$ ，将输出映射到 $Z \in R^{n\times C_o}$ 。全连接层 $\hat{v}=Zw+b$ ，其中 $w \in R^{C_o}$ ，于是输出 $\hat{v} \in R^n$ 。

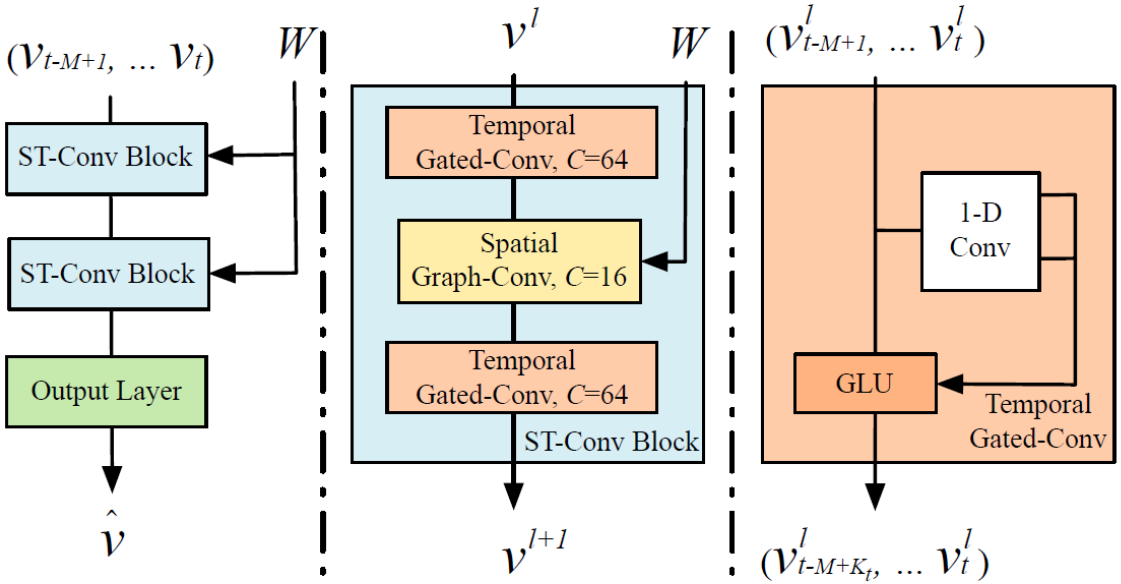
# STGCN-动态交通流量图预测

## 训练

- 损失函数是预测值和真实值的距离度量:

$$L\left(\hat{v}; W_\theta\right) = \left\|\hat{v}\left(v_{t-M+1}, \cdots, v_t, W_\theta\right) - v_{t+1}\right\|^2$$

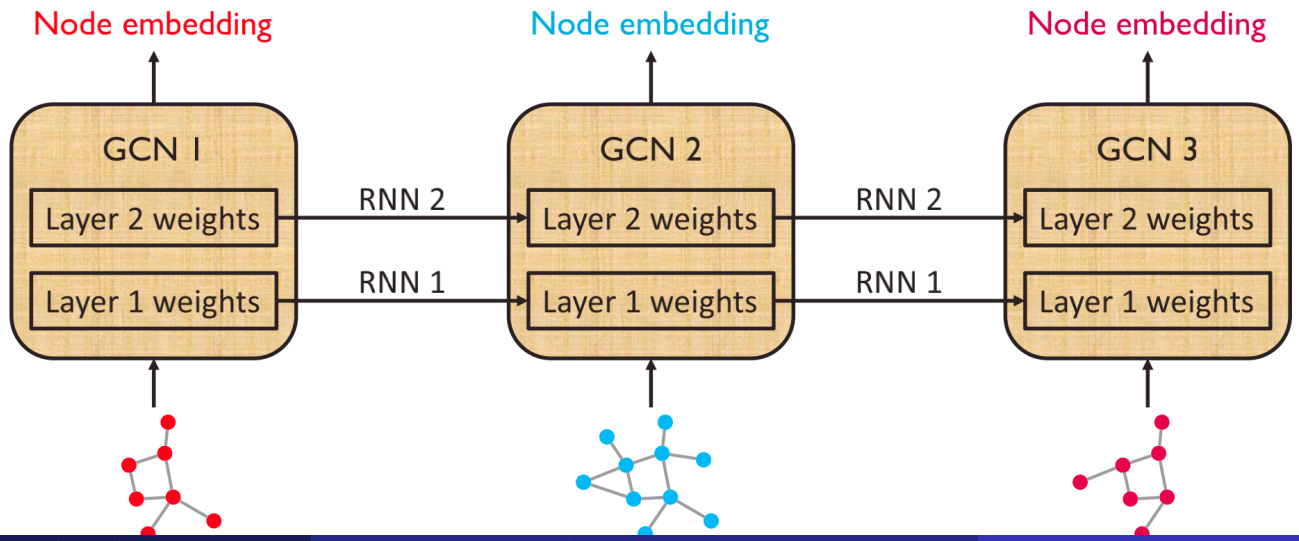
其中 $W_\theta$ 是所有可训练参数，$\hat{v}$ 是预测值，$v_{t+1}$ 是真实值。



# Evolvegcn: Evolving graph convolutional networks for dynamic graphs[7]

## 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

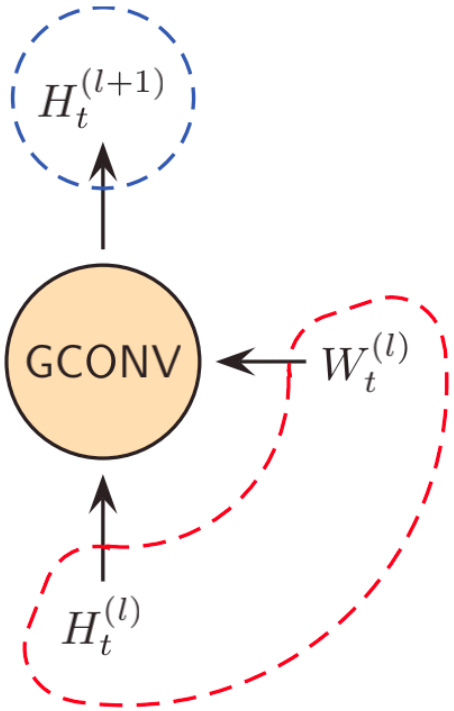
## 核心思想

- 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：

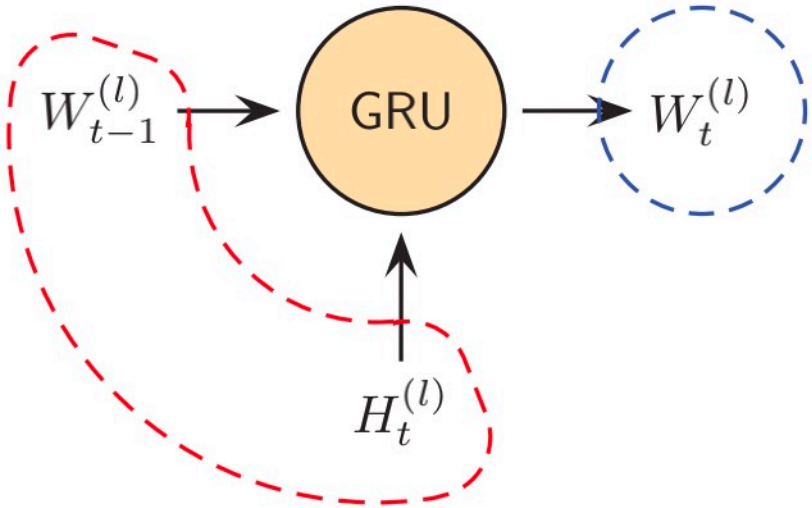
- EvolveGCN-H：将 $W_t^l$ 作为序列数据的隐藏状态，使用 GRU 来更新 $W_t^l$。
- EvolveGCN-O：将 $W_t^l$ 作为序列数据的当前输出，也作为下一时刻的输入，通过 LSTM 来建模。

# EvolveGCN-H

## EvolveGCN-H

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

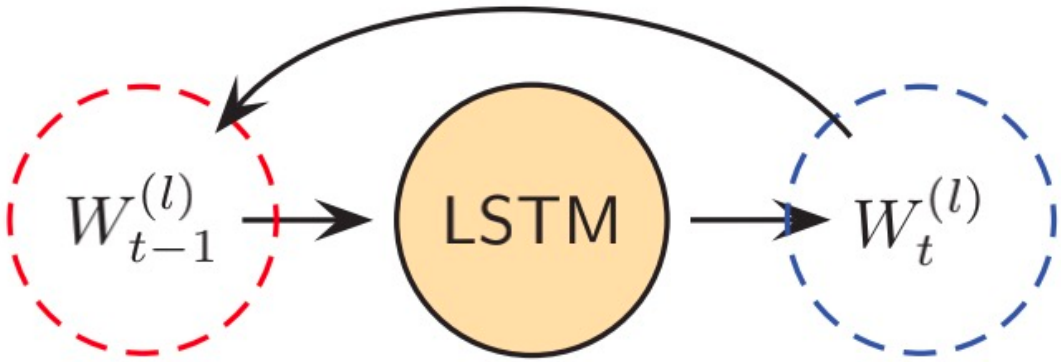
$$\overbrace{W_t^l}^{\text{GCN weights}} = \overbrace{W_t^l}^{\text{hidden state}} = \mathbf{GRU}\Big(\overbrace{H_t^l}^{\text{node embeddings}}, \overbrace{W_{t-1}^l}^{\text{GCN weights}}\Big)$$



## 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。

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



## 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）。

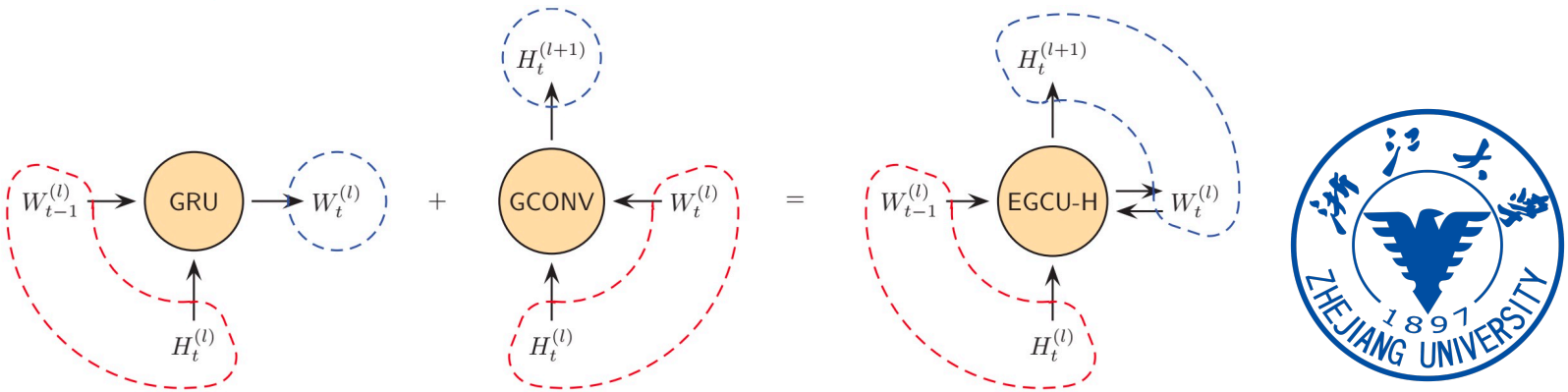
- EGCU-H：

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

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

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

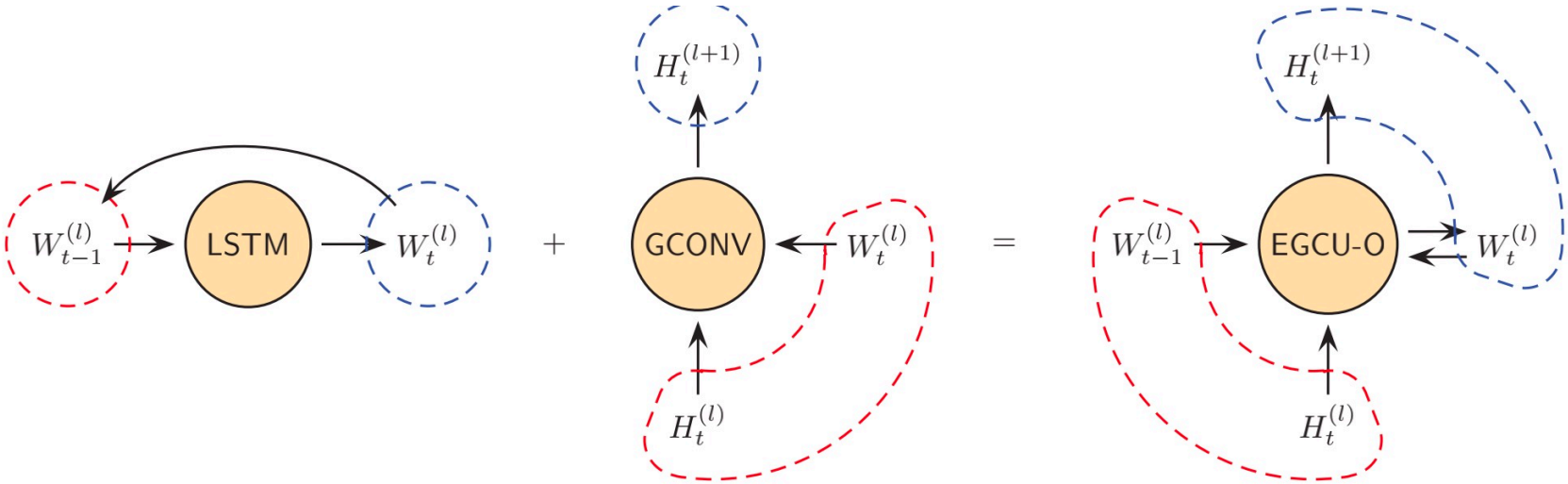
**end function**



# 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:

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

EvolveGCN-O:

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

- 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。

# 动态图神经网络发展前景

数据

1. Discrete Dynamic Graph
2. Continuous Dynamic Graph

模型

1. 纯图神经网络更新
2. Transformer

场景

1. 异常检测
2. 推荐系统
3. ...

# 课程总结



# 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.


# References II


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.

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.


# References III



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.