

图神经网络理论与应用

初探图神经网络

授课教师：周晟

浙江大学 软件学院

2024.11.12



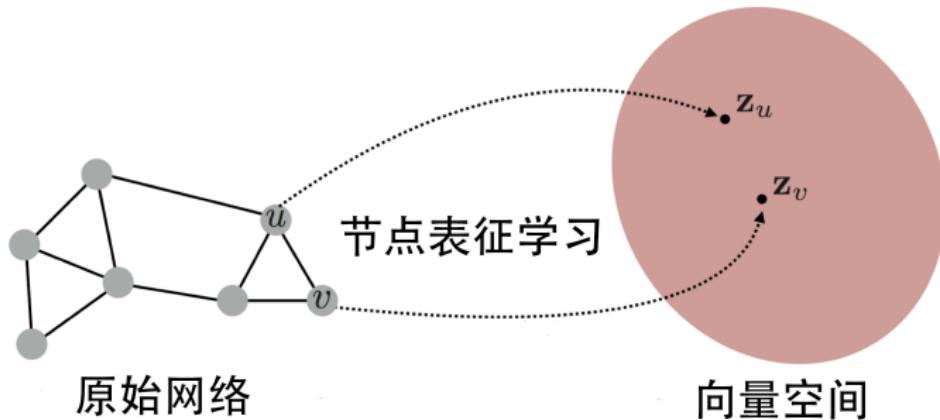
上节回顾

① 基于矩阵分解的方法

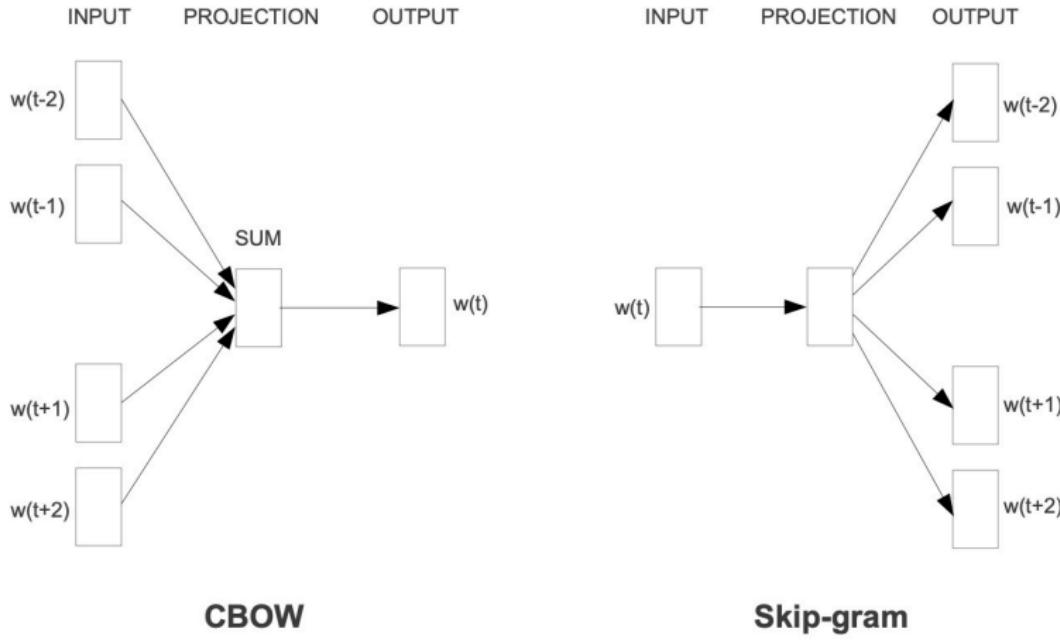
- ① LINE
- ② HOPE

② 基于随机游走的方法

- ① DeepWalk
- ② Node2Vec



Word2Vec



CBOW 和 Skip-gram

Word2Vec

从成对出现的词的角度，word2vec 的目标是最大化成对出现的概率：

$$P(D = 1 \mid w, c) = \sigma(\vec{w} \cdot \vec{c}) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{c}}}$$

$$P(D = 0 \mid w, c) = \sigma(-\vec{w} \cdot \vec{c}) = \frac{1}{1 + e^{\vec{w} \cdot \vec{c}}}$$

如果词汇量非常大，那么网络最后的 softmax 将会带来很大的时间开销。Word2Vec 提出了两种加快训练速度的方式：

- Hierarchical softmax
- Negative Sampling



随机游走与矩阵分解的等价性

SGNS 算法的目标函数为：

$$\ell = \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) (\log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbb{E}_{c_N \sim P_D} [\log \sigma(-\vec{w} \cdot \vec{c}_N)])$$

可以拆解为：

$$\begin{aligned}\ell &= \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) (\log \sigma(\vec{w} \cdot \vec{c})) \\ &\quad + \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) (k \cdot \mathbb{E}_{c_N \sim P_D} [\log \sigma(-\vec{w} \cdot \vec{c}_N)]) \\ &= \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) (\log \sigma(\vec{w} \cdot \vec{c})) \\ &\quad + \sum_{w \in V_W} \#(w) (k \cdot \mathbb{E}_{c_N \sim P_D} [\log \sigma(-\vec{w} \cdot \vec{c}_N)])\end{aligned}$$



随机游走与矩阵分解的等价性

其中数学期望可以通过观测样本进行估计：

$$\begin{aligned}\mathbb{E}_{c_N \sim P_D} [\log \sigma(-\vec{w} \cdot \vec{c}_N)] &= \sum_{c_N \in V_C} \frac{\#(c_N)}{|D|} \log \sigma(-\vec{w} \cdot \vec{c}_N) \\ &= \frac{\#(c)}{|D|} \log \sigma(-\vec{w} \cdot \vec{c}) + \sum_{c_N \in V_C \setminus \{c\}} \frac{\#(c_N)}{|D|} \log \sigma(-\vec{w} \cdot \vec{c}_N)\end{aligned}$$

对于观测到的一组词，其在优化过程中的目标为：

$$\ell(w, c) = \#(w, c) \log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \#(w) \cdot \frac{\#(c)}{|D|} \log \sigma(-\vec{w} \cdot \vec{c})$$



随机游走与矩阵分解的等价性

为了计算方便，令 $x = \vec{w} \cdot \vec{c}$ ，对于观测到的一组词，其优化的导数为：

$$\frac{\partial \ell}{\partial x} = \#(w, c) \cdot \sigma(-x) - k \cdot \#(w) \cdot \frac{\#(c)}{|D|} \cdot \sigma(x)$$

另导数等于 0 可得：

$$e^{2x} - \left(\frac{\#(w, c)}{k \cdot \#(w) \cdot \frac{\#(c)}{|D|}} - 1 \right) e^x - \frac{\#(w, c)}{k \cdot \#(w) \cdot \frac{\#(c)}{|D|}} = 0$$

另 $y = e^{2x}$ 将式子转化为 y 的二项式，易得 y 的解为 -1 和：

$$y = \frac{\#(w, c)}{k \cdot \#(w) \cdot \frac{\#(c)}{|D|}} = \frac{\#(w, c) \cdot |D|}{\#w \cdot \#(c)} \cdot \frac{1}{k}$$



随机游走与矩阵分解的等价性

将 x, y 代入之后可得：

$$\vec{w} \cdot \vec{c} = \log \left(\frac{\#(w, c) \cdot |D|}{\#(w) \cdot \#(c)} \cdot \frac{1}{k} \right) = \log \left(\frac{\#(w, c) \cdot |D|}{\#(w) \cdot \#(c)} \right) - \log k$$

其中 $\log \left(\frac{\#(w, c) \cdot |D|}{\#(w) \cdot \#(c)} \right)$ 是 pointwise mutual information (PMI)，最终词向量的计算等价为：

$$M_{ij}^{\text{SGNS}} = W_i \cdot C_j = \vec{w}_i \cdot \vec{c}_j = PMI(w_i, c_j) - \log k$$

即对 PMI 矩阵进行矩阵分解。



上节回顾

经典图表征学习算法与矩阵分解的等价性¹:

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

Notations in DeepWalk and LINE are introduced below. See detailed notations for PTE and node2vec in Section 2.

$\mathbf{A}: \mathbf{A} \in \mathbb{R}_+^{|\mathcal{V}| \times |\mathcal{V}|}$ is G 's adjacency matrix with $\mathbf{A}_{i,j}$ as the edge weight between vertices i and j ;

$D_{\text{col}}: D_{\text{col}} = \text{diag}(\mathbf{A}^\top \mathbf{e})$ is the diagonal matrix with column sum of \mathbf{A} ;

$D_{\text{row}}: D_{\text{row}} = \text{diag}(\mathbf{A}\mathbf{e})$ is the diagonal matrix with row sum of \mathbf{A} ;

D : For undirected graphs ($\mathbf{A}^\top = \mathbf{A}$), $D_{\text{col}} = D_{\text{row}}$. For brevity, D represents both D_{col} & D_{row} .

$D = \text{diag}(d_1, \dots, d_{|\mathcal{V}|})$, where d_i represents generalized degree of vertex i ;

$\text{vol}(G): \text{vol}(G) = \sum_i \sum_j \mathbf{A}_{i,j} = \sum_i d_i$ is the volume of a weighted graph G ;

T & b : The context window size and the number of negative sampling in skip-gram, respectively.

¹Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec(WSDM18)

Shallow Network Embedding

通过设定优化目标直接学习节点的表征（Embedding）：

$$\mathbf{H}^* = \arg \min \sum_{i=1}^N f(h_i)$$

Shallow Embedding 的缺陷²：

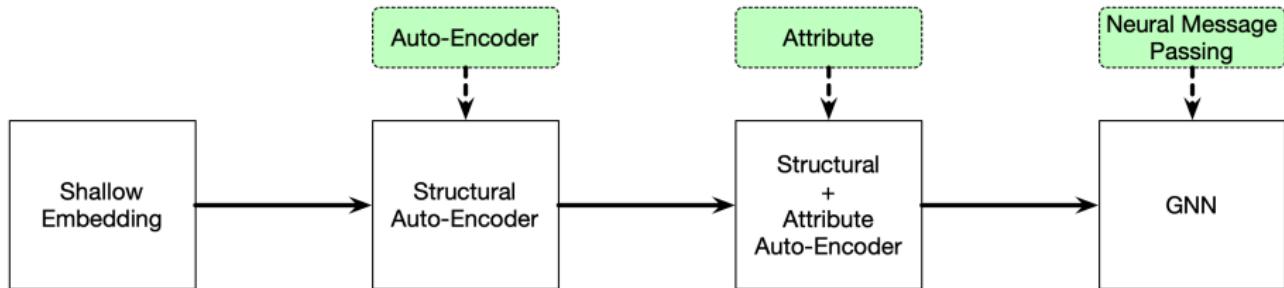
- ① 仅用于拟合节点的临近信息，无法拟合其他结构信息
- ② 无法融入属性等其他信息
- ③ 难以支持批量训练

²Representation Learning on Graphs: Methods and Applications

深度节点表征学习

使用深度学习进行节点表征学习的优势：

- ① 捕获节点间的非线性关系
- ② 捕获网络复杂结构关系
- ③ 较快的训练速度
- ④ ...



图神经网络发展历史

① 基于深度自编码器的节点表征学习

② 融合节点属性的节点表征学习

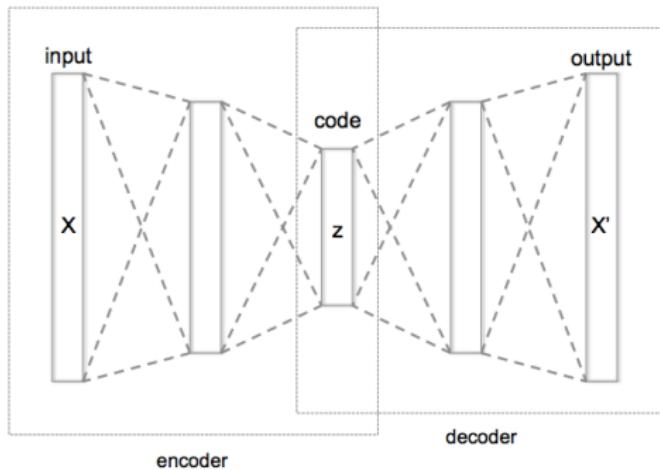
③ 基于神经消息传递的节点表征学习



深度自编码器

深度自编码器

深度自编码器（Deep AutoEncoder）通过编码器（Encoder）对数据进行压缩，通过解码器（Decoder）对数据进行解压，实现数据的有效降维。



深度自编码器

深度自编码器主要包括：

① 编码器

$$\mathbf{z} = \phi_e(\mathbf{x}; \Theta_e)$$

② 解码器

$$\hat{\mathbf{x}} = \phi_d(\mathbf{z}; \Theta_d)$$

③ 优化目标

$$\{\Theta_e^*, \Theta_d^*\} = \arg \min_{\Theta_e, \Theta_d} \sum_{\mathbf{x} \in X} \|\mathbf{x} - \phi_d(\phi_e(\mathbf{x}; \Theta_e); \Theta_d)\|^2$$



深度自编码器

深度自编码器的特点：

- ① 简单易用，无需数据假设，无需先验知识
- ② 可以使用多种深度神经网络架构
- ③ 捕获数据集的全局特性
- ④ 训练速度快
- ⑤ ...

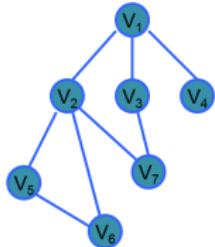
深度自编码器在计算机视觉，自然语言处理等领域已取得广泛的应用和成功。

使用深度自编码器提取网络中节点的特征？

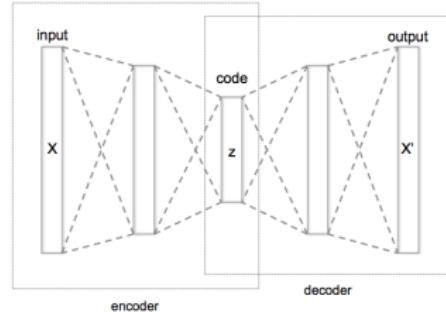


基于深度自编码器的节点表征学习

一种简单的基于深度自编码器的节点表征学习模型：

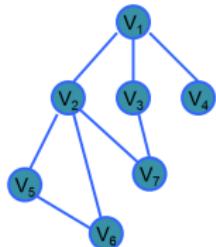


	0	1	2	3	4	5	6
0	0	1	1	1	0	0	0
1	1	0	0	0	1	1	1
2	1	0	0	0	0	0	1
3	1	0	0	0	0	0	0
4	0	1	0	0	0	1	0
5	0	1	0	0	1	0	0
6	0	1	1	0	0	0	0

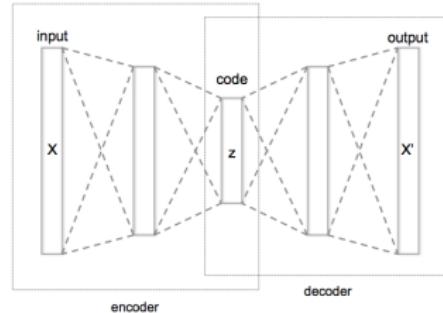


基于深度自编码器的节点表征学习

一种简单的基于深度自编码器的节点表征学习模型：



	0	1	2	3	4	5	6
0	0	1	1	1	0	0	0
1	1	0	0	0	1	1	1
2	1	0	0	0	0	0	1
3	1	0	0	0	0	0	0
4	0	1	0	0	0	1	0
5	0	1	0	0	1	0	0
6	0	1	1	0	0	0	0

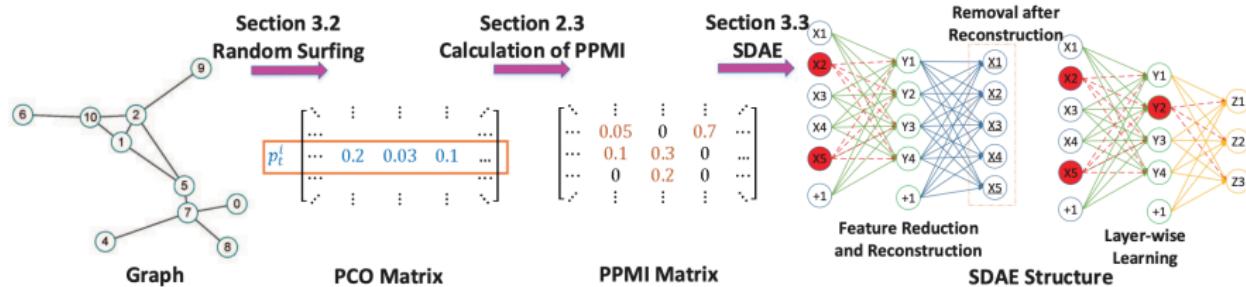


缺陷

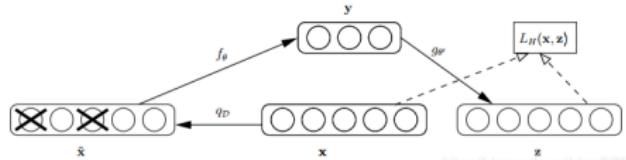
- ① 模型容易坍塌
- ② 无法捕获高阶结构特征
- ③ ...



被编码数据的改进



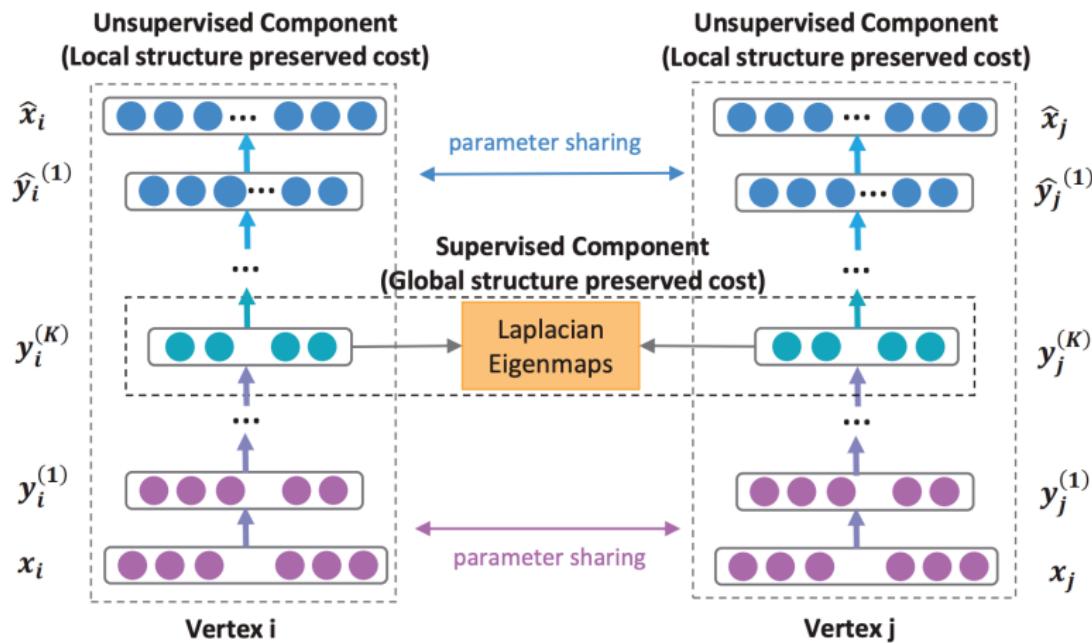
SDAE³: 用 PPMI 矩阵代替原始邻接矩阵



Denoised Auto-Encoder

³Deep Neural Networks for Learning Graph Representations(AAAI-16)

结构和属性的统一建模



Structural Deep Network Embedding(KDD 2016)[Wang et al., 2016]

Structural Deep Network Embedding

研究动机

- ① 节点表征需要同时捕获节点间的低阶和高阶邻近性
- ② 网络中的节点间依赖关系非线性
- ③ 网络结构稀疏

解决方案

- ① 使用深度神经网络捕获节点间的非线性依赖关系
- ② 节点表征同时受到一阶二阶临近关系的约束
- ③ 对节点间是否有边施加不同的约束

Structural Deep Network Embedding

自编码器设计：

$$\mathbf{y}_i^{(1)} = \sigma(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)})$$

$$\mathbf{y}_i^{(k)} = \sigma(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), k = 2, \dots, K$$

自编码器约束：

$$\mathcal{L}_{1st} = \sum_{i,j=1}^n s_{i,j} \left\| \mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)} \right\|_2^2 = \sum_{i,j=1}^n s_{i,j} \left\| \mathbf{y}_i - \mathbf{y}_j \right\|_2^2$$

节点表征拟合一阶临近关系 (Proximity)

Structural Deep Network Embedding

稀疏向量的自编码器约束：

$$\mathcal{L}_{2nd} = \sum_{i=1}^n \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b}_i\|_2^2 = \|(\hat{X} - X) \odot B\|_F^2$$

$$b_{ij} = \begin{cases} 1 & x_{ij} = 0 \\ \beta > 1 & x_{ij} = 1 \end{cases}$$

SDNE 的优化目标：

$$\begin{aligned}\mathcal{L}_{mix} &= \mathcal{L}_{2nd} + \alpha \mathcal{L}_{1st} + \nu \mathcal{L}_{reg} \\ &= \|(\hat{X} - X) \odot B\|_F^2 + \alpha \sum_{i,j=1}^n s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 + \nu \mathcal{L}_{reg} \\ \mathcal{L}_{reg} &= \frac{1}{2} \sum_{k=1}^K \left(\|W^{(k)}\|_F^2 + \|\hat{W}^{(k)}\|_F^2 \right)\end{aligned}$$



Structural Deep Network Embedding

SDNE 模型总结：

优点

- ① 首次使用深度自编码器用于节点表征学习
- ② 显式克服自编码器处理稀疏特征的困扰
- ③ 融合图特性的约束
- ④ 模型简单有效

缺点

- ① 只能捕获低阶结构特征
- ② 仅考虑节点结构特征
- ③ 复杂度高

其他结构表征学习

Category	Source of raw features
Local structure	Degree/in-degree/out-degree [70]
	Neighbors/in-degree neighbors/out-degree neighbors [81]
	Node state [41]
	Adjacent edge state [49, 89]
Global structure	Multi-hop neighborhoods [105]
	Node community membership [75, 90]
	Node connectivity pattern [4, 36]
	Node degree distribution [29]
	Node rank [53]
	Node identity [70, 84]

常见的结构特征⁴

⁴Network Representation Learning: From Preprocessing, Feature Extraction to Node Embedding

① 基于深度自编码器的节点表征学习

② 融合节点属性的节点表征学习

③ 基于神经消息传递的节点表征学习



研究背景

研究背景

矩阵分解，随机游走等方法仅能捕获网络的结构特征，而真实的网络中节点往往带有属性，这些属性可以提供每个节点独有的特性。好的节点表征学习应当同时捕获网络结构（Topology）特征和节点属性（Attribute）特征。



计算机网络



社交网络



交通网络

属性与结构的统一建模

属性与结构一起建模的**意义**:

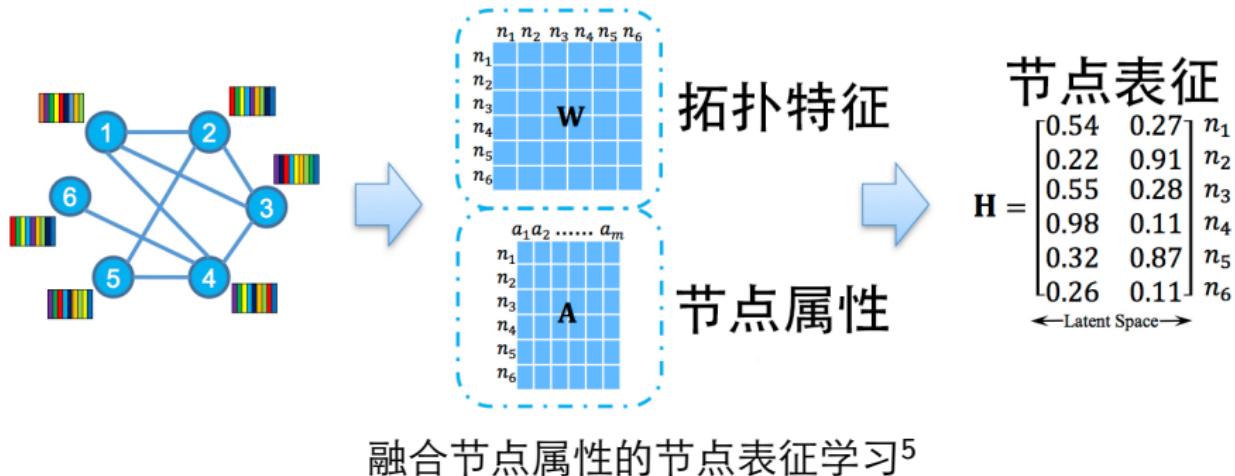
- ① 缺失信息补充
- ② 结构信息降噪
- ③ 克服稀疏性问题
- ④ 刻画图数据特点

带属性网络建模的**挑战**:

- ① 节点属性如何建模?
- ② 节点属性与图结构之间是什么关系?
- ③ 节点属性与图结构信息如何融合?



融合节点属性的节点表征学习

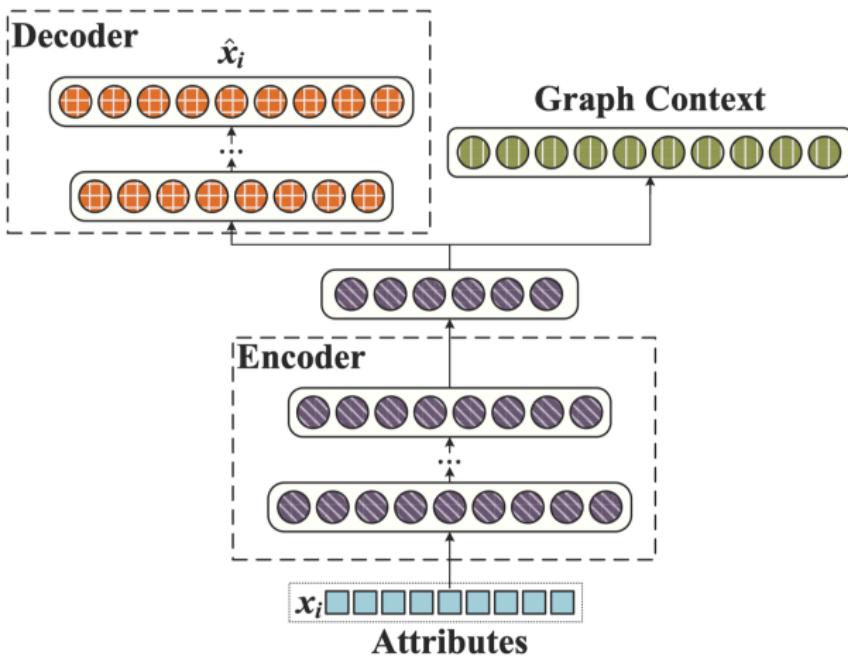


$$\mathcal{L} = \|\mathbf{S} - \mathbf{H}\mathbf{H}^T\|_F^2 + \lambda \sum_{(i,j) \in \mathcal{E}} w_{ij} \|h_i - h_j\|_2$$



⁵ Accelerated Attributed Network Embedding(SDM 17)

ANRL: Attributed Network Representation Learning via Deep Neural Networks



ANRL: Attributed Network Representation Learning via Deep Neural Networks

研究动机

- ① 自编码器擅长提取连续的属性特征
- ② 自编码器难以编码稀疏的拓扑结构信息
- ③ 节点属性和网络拓扑结构难以融合
- ④ 节点邻居与节点有相似的特性，并且可以减少单个节点学习时的噪声

研究方案

- ① 使用自编码器编码节点属性
- ② 使用属性信息拟合网络结构
- ③ 拟合节点和邻居的相似性

ANRL: Attributed Network Representation Learning via Deep Neural Networks

编码器设计：

$$\begin{aligned}\mathbf{y}_i^{(1)} &= \sigma(\mathbf{W}^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}) \\ \mathbf{y}_i^{(k)} &= \sigma(\mathbf{W}^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)})\end{aligned}$$

编码器优化目标：

$$\mathcal{L}_{ae} = \sum_{i=1}^n \|\hat{\mathbf{x}}_i - T(v_i)\|_2^2$$



ANRL: Attributed Network Representation Learning via Deep Neural Networks

邻居特征整合函数 $T(v_i)$:

- ① Weighted Average Neighbor.

$$T(v_i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{x}_j$$

- ② Elementwise Median Neighbor.

$$\tilde{x}_k = \text{Median} \left(w_{i1} \mathbf{x}_{1k}, w_{i2} \mathbf{x}_{2k}, \dots, w_{i|\mathcal{N}(i)|} \mathbf{x}_{|\mathcal{N}(i)|k} \right)$$

早期基于邻域的节点重构思想
后被广泛应用于 GNN 设计和图异常检测场景



ANRL: Attributed Network Representation Learning via Deep Neural Networks

融合网络结构和节点特征的重构损失：Attribute-aware Skip-gram Model

$$\mathcal{L}_{sg} = - \sum_{i=1}^n \sum_{c \in C} \sum_{-b \leq j \leq b, j \neq 0} \log p(v_{i+j} | \mathbf{x}_i)$$

$$p(v_{i+j} | \mathbf{x}_i) = \frac{\exp(\mathbf{v}'^T_{i+j} f(\mathbf{x}_i))}{\sum_{v=1}^n \exp(\mathbf{v}'^T_v f(\mathbf{x}_i))}$$

利用属性重构高阶邻居信息



ANRL: Attributed Network Representation Learning via Deep Neural Networks

ANRL 目标函数：

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{sg} + \alpha \mathcal{L}_{ae} + \beta \mathcal{L}_{reg} \\ &= - \sum_{i=1}^n \sum_{c \in C} \sum_{-b \leq j \leq b, j \neq 0} \log \frac{\exp \left(\mathbf{u}_{i+j}^T \mathbf{y}_i^{(K)} \right)}{\sum_{v=1}^n \exp \left(\mathbf{u}_v^T \mathbf{y}_i^{(K)} \right)} \\ &\quad + \alpha \sum_{i=1}^n \|\hat{\mathbf{x}}_i - T(v_i)\|_2^2 + \frac{\beta}{2} \sum_{k=1}^K \left(\|\mathbf{W}^{(k)}\|_F^2 + \|\hat{\mathbf{W}}^{(k)}\|_F^2 \right)\end{aligned}$$



ANRL: Attributed Network Representation Learning via Deep Neural Networks

Datasets	Citeseer		Pubmed		Fraud Detection	
Evaluation	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
SVM	0.667	0.626	0.856	0.855	0.725	0.719
autoencoder	0.630	0.565	0.792	0.800	0.732	0.726
DeepWalk	0.583	0.534	0.809	0.795	0.509	0.464
node2vec	0.607	0.561	0.815	0.802	0.571	0.519
LINE	0.542	0.512	0.766	0.749	0.659	0.654
SDNE	0.569	0.528	0.699	0.677	0.662	0.656
AANE	0.579	0.541	0.784	0.765	0.654	0.643
SNE	0.632	0.615	0.803	0.797	0.662	0.654
Planetoid-T	0.656	0.594	0.851	0.847	0.692	0.693
TriDNR	0.633	0.587	0.843	0.824	0.686	0.685
SEANO	0.713	0.662	0.859	0.848	0.703	0.704
ANRL-OWN	0.652	0.606	0.842	0.845	0.724	0.720
ANRL-EMN	0.716	0.668	0.865	0.867	0.733	0.731
ANRL-WAN	0.729	0.673	0.876	0.871	0.759	0.755

ANRL 实验结果

PRRE: Personalized Relation Ranking Embedding for Attributed Networks

研究动机 1

节点之间的关系可以通过拓扑结构和节点属性衡量，然而两者并不是完全一致，存在个性（Individuality），不能简单地合并。

研究动机 2

虽然拓扑结构和节点属性存在差异性，但是两者都是描述节点间关系的度量，因此存在共性（Commonality），不能完全分开处理。

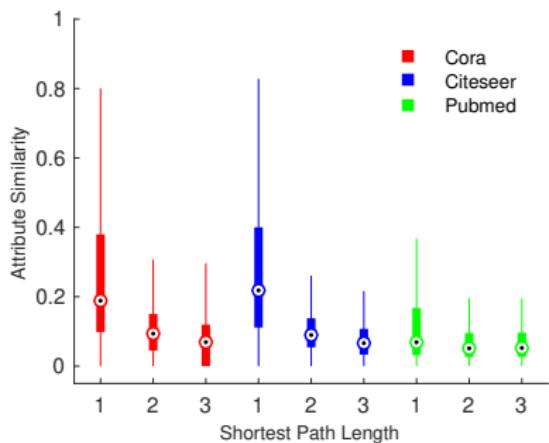
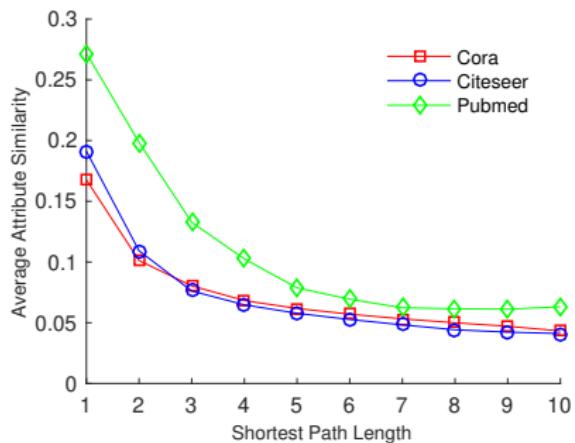
现象：

- ① 有相似的属性的节点在网络上距离很远
- ② 网络上相连的节点属性完全不同

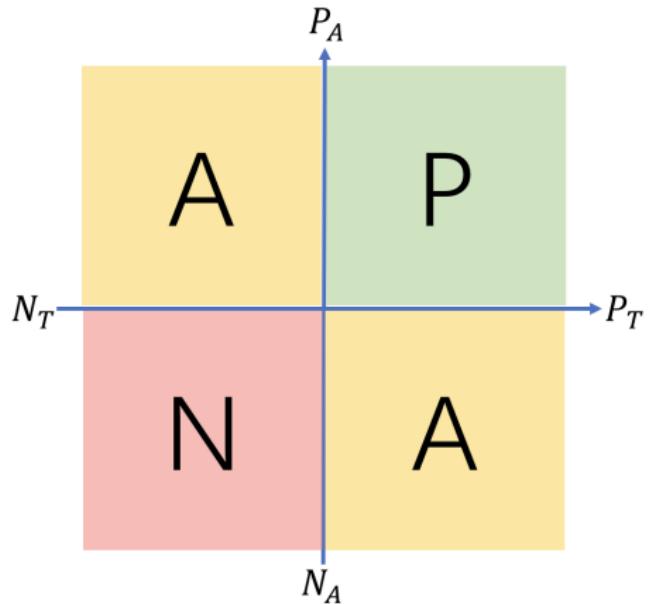


PRRE: Personalized Relation Ranking Embedding for Attributed Networks

真实数据集上节点属性和拓扑结构相关性分析



PRRE: Personalized Relation Ranking Embedding for Attributed Networks



基于拓扑和属性的节点关系划分：

- ① Positive: 节点属性和拓扑均相似
- ② Ambiguous: 节点属性或拓扑相似
- ③ Negative: 节点属性和拓扑均不相似



PRRE: Personalized Relation Ranking Embedding for Attributed Networks

节点排序性质

- ① **Totality:** $\forall v_i, v_j, v_k \in \mathcal{V} : j \neq k \Rightarrow j \geq_i k \vee k \geq_i j$
- ② **Antisymmetry:** $\forall v_i, v_j, v_k \in \mathcal{V} : j \geq_i k \wedge k \geq_i j \Rightarrow j = k$
- ③ **Transitivity:** $\forall v_i, v_j, v_k, v_l \in \mathcal{V} : j \geq_i k \wedge k \geq_i l \Rightarrow j \geq_i l$

节点关系排序：

$$\prod_{\{i,j,k\} \in \mathcal{V}} P(j \geq_i k).$$

节点相似性：

$$\sigma_{ij} = \sigma(h_i^T h_j) = \frac{1}{1 + e^{-h_i^T h_j}},$$

PRRE: Personalized Relation Ranking Embedding for Attributed Networks

节点的关系通过阈值 θ 判为 Positive/Negative，进而在融合拓扑结构和节点属性之后判为三大类：Positive / Ambiguous / Negative。

阈值质量

好的阈值应该将所有数据尽可能分开，且阈值可以进行学习。

$$g(\theta_T) = (\overline{S_T^P} - \theta_T)(\theta_T - \overline{S_T^N}),$$

$$g(\theta_A) = (\overline{S_A^P} - \theta_A)(\theta_A - \overline{S_A^N}),$$

$$g(\theta_T, \theta_A) = (\overline{S_T^P} - \theta_T)(\theta_T - \overline{S_T^N}) + (\overline{S_A^P} - \theta_A)(\theta_A - \overline{S_A^N}),$$

PRRE: Personalized Relation Ranking Embedding for Attributed Networks

节点的表征和阈值可以在统一的框架下学习：

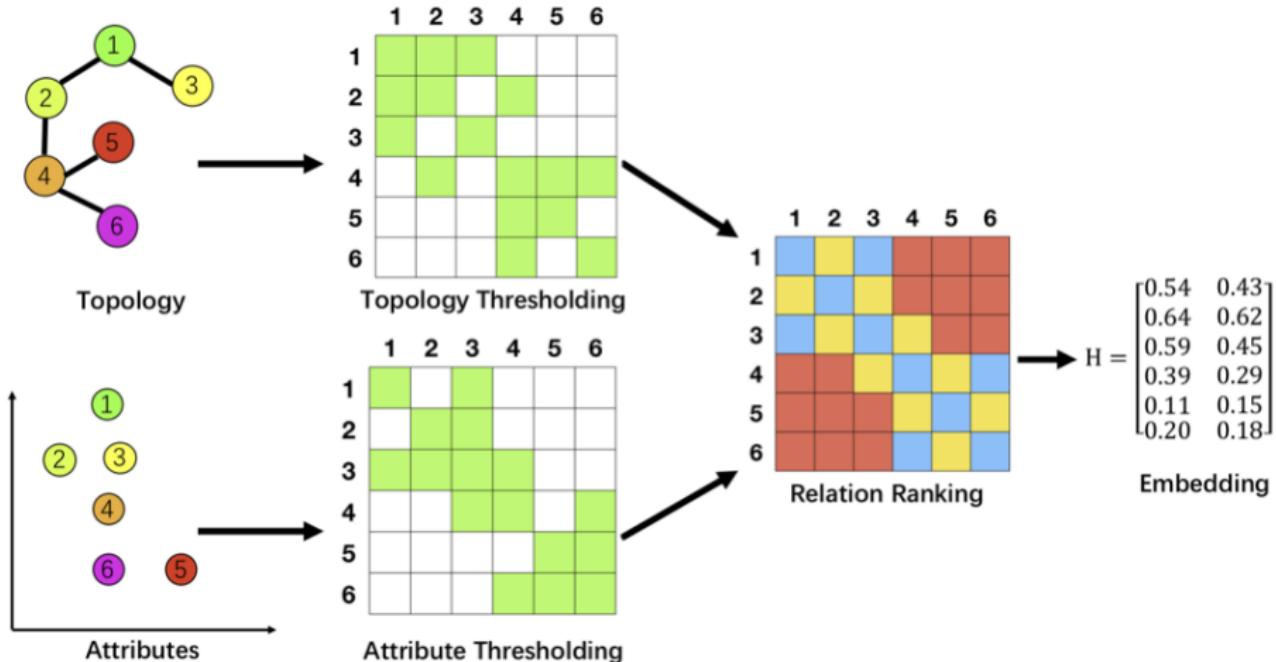
$$\prod_{\{i,j,k\} \in \mathcal{V}} P(j \geq_i k | \theta, H) = \left[\frac{\sigma_{ij} - \sigma_{ik} + 1}{2} \right]^{\frac{1}{1+g(\theta)}},$$

最终优化目标：

$$\begin{aligned} \mathcal{J}(H, \theta_T, \theta_A) = & \prod_{i \in \mathcal{V}} \left(\prod_{p \in P} \prod_{a \in A} P(p \geq_i a | \theta_A, \theta_T, H) \right. \\ & \left. \prod_{a \in A} \prod_{n \in N} P(a \geq_i n | \theta_A, \theta_T, H) \right) \end{aligned}$$



PRRE: Personalized Relation Ranking Embedding for Attributed Networks

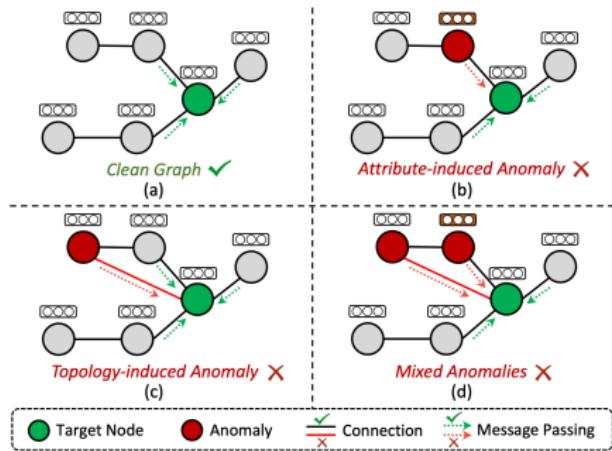


PRRE: Personalized Relation Ranking Embedding for Attributed Networks

总结

- ① 研究目标：学习能够融合节点属性和拓扑结构的节点表征
- ② 研究动机：节点的属性和拓扑特征在融合时存在分歧
- ③ 研究方案：融合节点的属性和拓扑结构进行三元关系排序：
 $\text{Positive} > \text{Ambiguous} > \text{Negative}$ ，并让节点表征保持关系排序

Guarding Graph Neural Networks for Unsupervised Graph Anomaly Detection

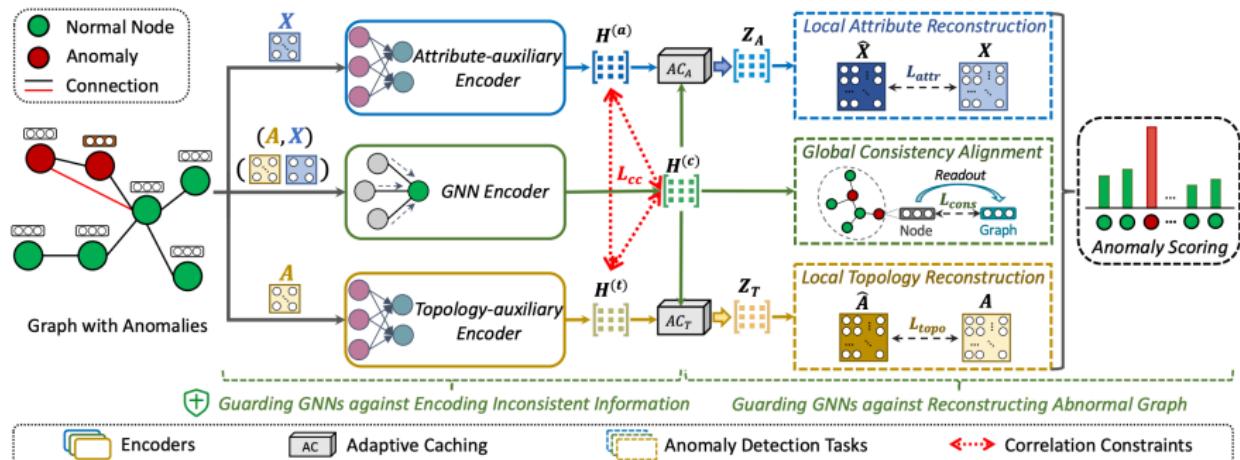


异常/噪声会对图上的结构/属性造成影响：

- ① 属性异常：影响基于属性相似度的关系拟合
- ② 边异常：影响基于拓扑临近性的关系拟合
- ③ 混合异常：两者同时影响



Guarding Graph Neural Networks for Unsupervised Graph Anomaly Detection



基于属性/结构解耦合的图高效表征

① 基于深度自编码器的节点表征学习

② 融合节点属性的节点表征学习

③ 基于神经消息传递的节点表征学习



经典节点表征学习方法的缺陷

- ① 没有满足图数据的**排列不变性**
- ② 学习过程没有**显式**基于图结构
- ③ 学习过程缺乏**可解释性**
- ④ ...

需要一个更“神经网络”的节点表征学习框架

研究背景

经典节点表征学习方法的缺陷

- ① 没有满足图数据的**排列不变性**
- ② 学习过程没有**显式**基于图结构
- ③ 学习过程缺乏**可解释性**
- ④ ...

需要一个更“神经网络”的节点表征学习框架

图神经网络！

- ① 同配性假设
- ② 排列不变性

同配性假设

同配性 (Homophily) 假设

同配性 (Homophily) 假设是指在网络中，相邻的节点往往会有相似的属性或标签

节点的同配性：

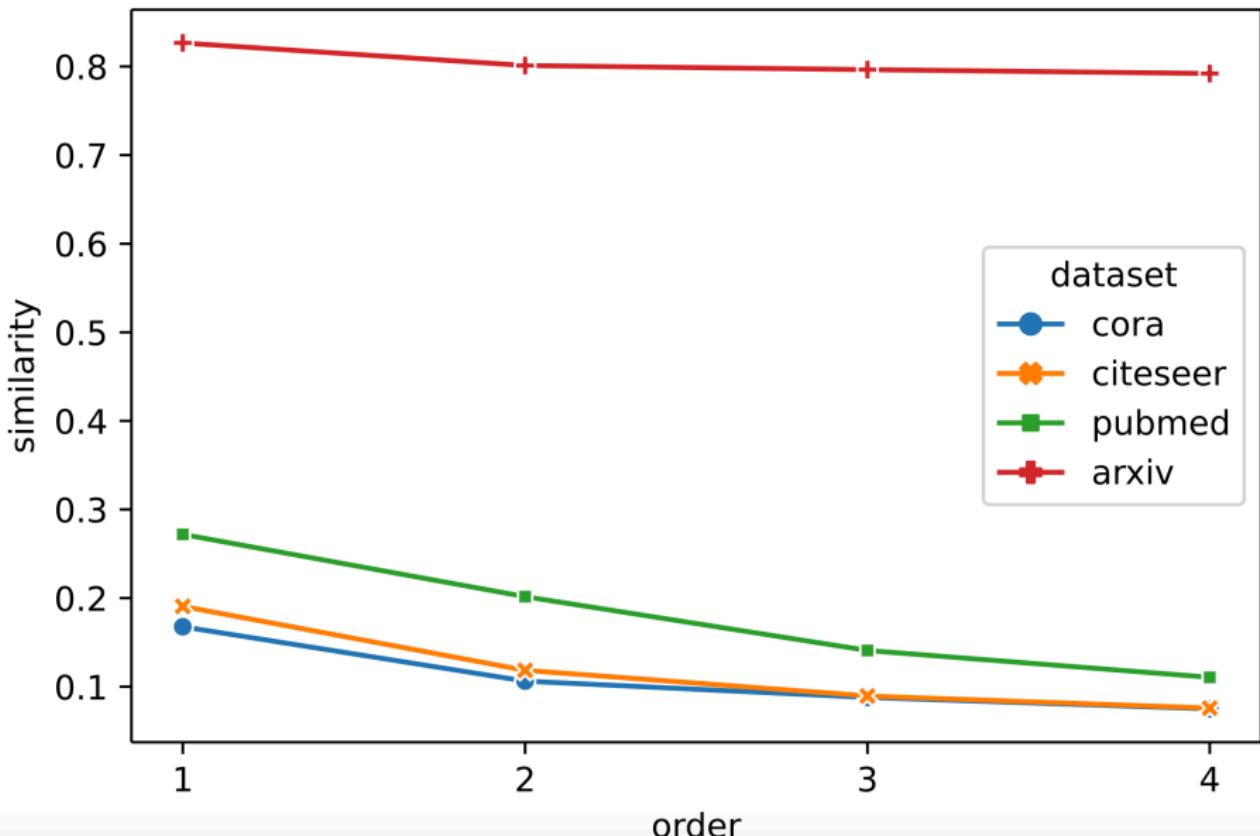
$$\mathcal{H}_{\text{node}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{u \in \mathcal{N}(v) : y_v = y_u\}|}{|\mathcal{N}(v)|}$$

边的同配性：

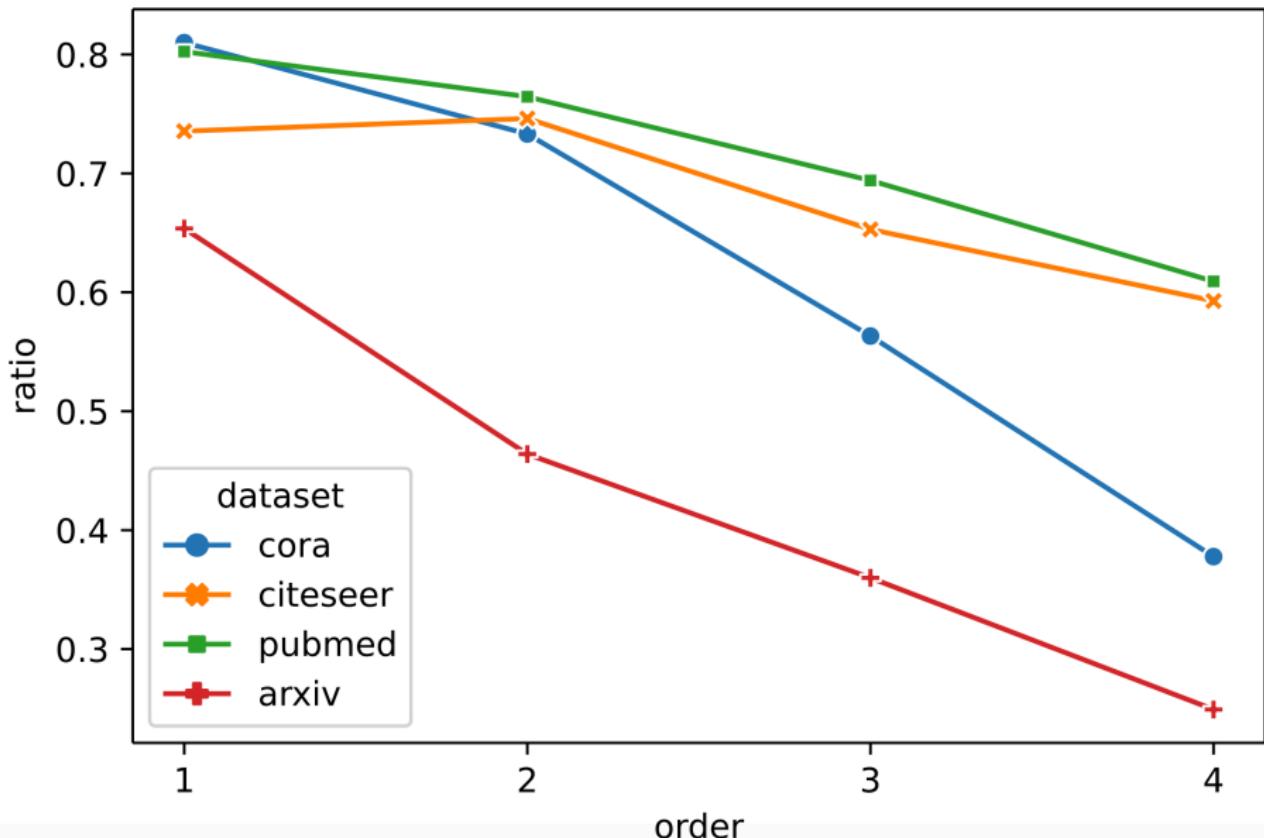
$$\mathcal{H}_{\text{edge}} = \frac{|\{(v, u) \in \mathcal{E} : y_v = y_u\}|}{|\mathcal{E}|}.$$



拓扑相似性与属性相似性



拓扑相似性与标签一致性



同配性假设

Types	Datasets	# Nodes	# Edges	# Features	# Classes	\mathcal{H}_{node}	\mathcal{H}_{edge}
WebKB Webpage	Cornell	183	295	1,703	5	0.11	0.30
	Texas	183	309	1,703	5	0.06	0.11
	Wisconsin	251	499	1,703	5	0.16	0.21
Author Co-occurrence	Actor	7,600	33,544	931	5	0.24	0.22
Wikipedia Webpage	Chameleon	2,277	36,101	2,325	5	0.25	0.23
	Squirrel	5,201	217,073	2,089	5	0.22	0.22
	Wiki	1,925,342	303,434,860	600	5	-	0.39
Citation	ArXiv-Year	169,343	1,166,243	128	5	-	0.22
	Snap-patents	2,923,922	13,975,788	269	5	-	0.07
Social Networks	Deezer-Europe	28,281	92,752	31,241	2	0.53	0.53
	Penn94	41,554	1,362,229	5	2	-	0.47
	Twitch-Gamers	168,114	6,797,557	7	2	-	0.55
	Genius	421,961	984,979	12	2	-	0.62
	Pokec	1,632,803	30,622,564	65	2	-	0.45
Webpage Review	YelpChi	45,954	3,846,979	32	2	0.77	0.77

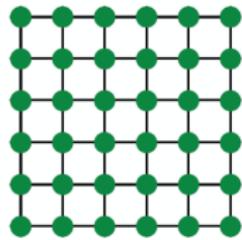
异配性 (Heterophily) 网络

网格结构数据

网格结构数据 (Grid Structure Data)

网格结构数据 (Grid Structure Data) 是指节点的特征按照网格形式排列，不同位置对应不同信息且位置之间不能交换。

常见的网格结构数据包括图像和序列数据。

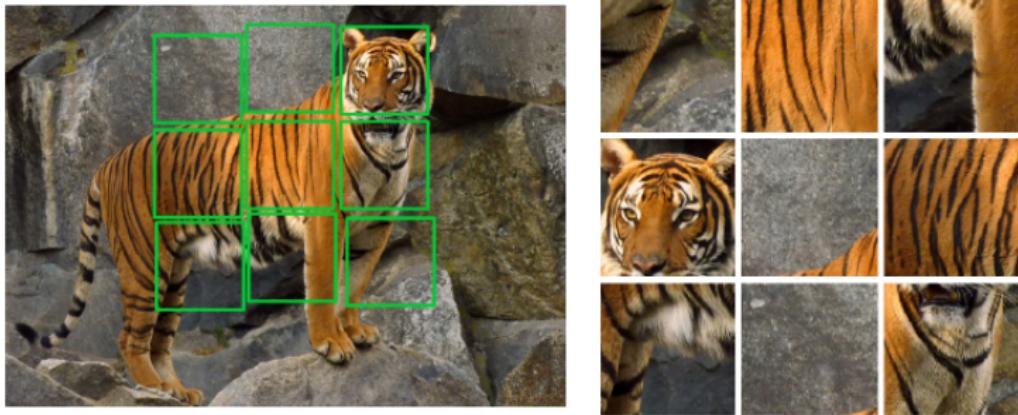


图数据的排列不变性

排列不变性 (Permutation Invariance)

对于网络数据来说，节点的排序编号不影响网络的结构，这种性质称为排列不变性 (Permutation Invariance)。

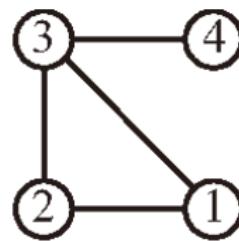
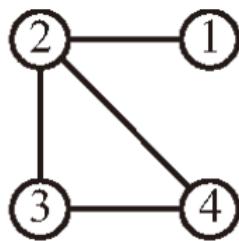
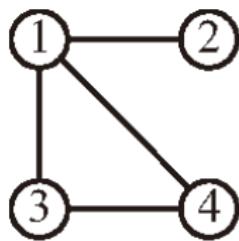
$$f(\mathbf{PAP}^\top) = f(\mathbf{A})$$
$$f(\mathbf{PAP}^\top) = \mathbf{P}f(\mathbf{A})$$



图数据的排列不变性

邻接矩阵的局限

如果使用邻接矩阵 A 的行向量 A_i 作为节点 v_i 的特征，则不满足排列不变性。



$$\begin{bmatrix} 0, 1, 1, 1 \\ 1, 0, 0, 0 \\ 1, 0, 0, 1 \\ 1, 0, 1, 0 \end{bmatrix}$$

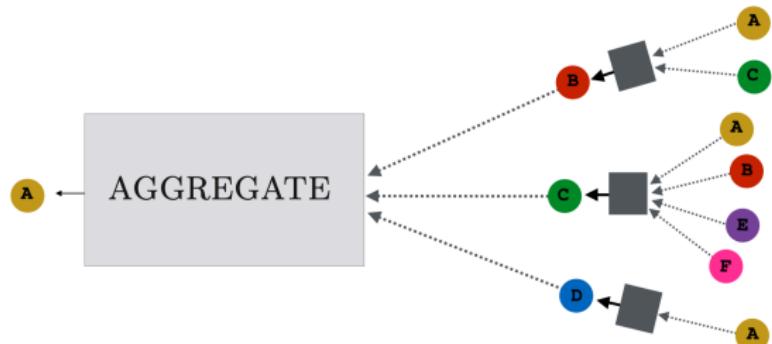
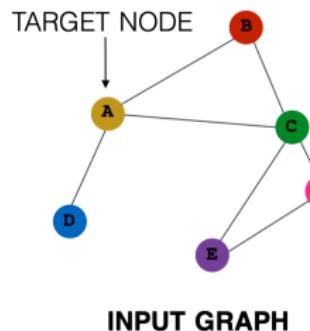
$$\begin{bmatrix} 0, 1, 0, 0 \\ 1, 0, 1, 1 \\ 0, 1, 0, 1 \\ 0, 1, 1, 0 \end{bmatrix}$$

$$\begin{bmatrix} 0, 1, 1, 0 \\ 1, 0, 1, 0 \\ 1, 1, 0, 1 \\ 0, 0, 1, 0 \end{bmatrix}$$

神经消息传递 (Neural Message Passing)

神经消息传递 (Neural Message Passing)

神经网络传递 (Neural Message Passing) 是指节点间传递特征信息并使用神经网络更新的一种机制。



神经消息传递 (Neural Message Passing)

神经消息传递的数学形式：

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)} \left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)} \left(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\} \right) \right) \\ &= \text{UPDATE}^{(k)} \left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} \right),\end{aligned}$$

核心模块：

- ① Aggregate
- ② Update

神经消息传递学习节点表征：

$$\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}$$



神经消息传递 (Neural Message Passing)

传递的消息是什么? Aggregate

- ① 节点以及周围节点的属性信息 (如果没有属性怎么办?)
- ② 节点以及周围节点的结构信息 (Degree)
- ③ 融合属性和结构的子图信息

传递的消息如何更新节点表征? Update

- ① 节点属性变换
- ② 邻居节点属性变换
- ③ 变换后的属性融合

神经消息传递的优势?

为什么用神经消息传递？

优势 1

仅依靠节点自身难以准确评估节点的模式，需要借助更多的信息

为什么不直接整合所有其他节点的信息？（Transformer）

优势 2

同配性假设的存在，使得节点周围存在具有相似模式（属性、标签）的邻居节点，这些节点可以为学习节点的特性提供帮助。

为什么不直接整合所有附近的邻居节点？

优势 3

单层邻居信息太少，多层邻居数量过多，且没有固定的邻居个数。考虑传播的方式，减少参数量。

基础 GNN 模型设计

基础 GNN 模型

节点的低维表征通过平均节点和邻居的属性得到

$$\mathbf{h}_u^{(k)} = \sigma \left(\mathbf{W}_{\text{self}}^{(k)} \mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)} \right)$$

- Aggregate: 一阶邻居的特征

$$\mathbf{m}_{\mathcal{N}(u)} = \text{AGGREGATE}^{(k)} \left(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\} \right)$$

- Update: 非线性激活

$$\text{UPDATE} \left(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)} \right) = \sigma \left(\mathbf{W}_{\text{self}} \mathbf{h}_u + \mathbf{W}_{\text{neigh}} \mathbf{m}_{\mathcal{N}(u)} \right)$$

通用 GNN 模块设计

Aggregation

- ① Neighbor Pooling
- ② Neighbor Normalization
- ③ Neighbor Attention

Update

- ① Concatenation
- ② Skip-Connection
- ③ Gated Update

Neighbor Pooling

研究动机

Neighbor Pooling 通过将邻居节点视作集合 (Set) 来保持置换不变性 (permutation invariant)，最常用的是 mean pooling (Homophily 假设)：

$$\mathbf{m}_{\mathcal{N}(u)} = \text{MLP}_\theta \left(\sum_{v \in N(u)} \text{MLP}_\phi (\mathbf{h}_v) \right)$$

所有将一个向量的集合作映射到一个向量的函数可以用上式拟合 [Zaheer et al., 2017]。

Neighbor Pooling

Pooling of permutation-sensitive function

利用 permutation-sensitive function，然后将不同 permutation 的结果求平均得到，通用的形式为：

$$\mathbf{m}_{\mathcal{N}(u)} = \text{MLP}_\theta \left(\frac{1}{|\Pi|} \sum_{\pi \in \Pi} \rho_\phi \left(\mathbf{h}_{v_1}, \mathbf{h}_{v_2}, \dots, \mathbf{h}_{v_{|\mathcal{N}(u)|}} \right)_{\pi_i} \right)$$

ρ_ϕ 通常是用于处理序列化数据的模块，如 RNN, GRU, LSTM 等

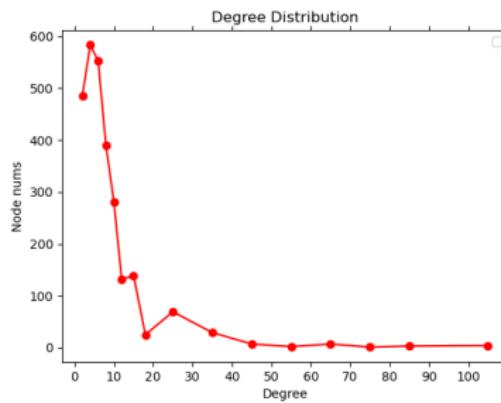
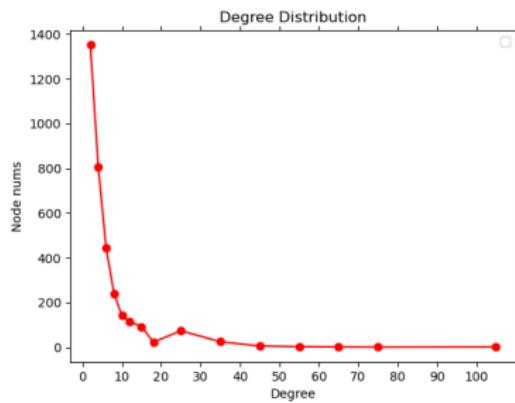
缺陷

- ① 计算复杂度高
- ② 序列缺乏实际意义

Neighbor Normalization

研究动机

网络中节点的度分布通常不均匀，不同度的邻居传递的信息量通常不同，将所有邻居节点同等对待容易引起信息丢失。



Neighbor Normalization

通过 Neighbor Normalization 将邻居节点进行区分，从而提升信息整合的效果：

$$\mathbf{m}_{\mathcal{N}(u)} = \frac{\sum_{v \in \mathcal{N}(u)} \mathbf{h}_v}{|\mathcal{N}(u)|}$$

为了保持消息传递的对称性，可以同时考虑节点自身的度：

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}$$

缺点

- ① 信息丢失
- ② 易受结构噪声影响
- ③ 仅在度分布不均衡时有效

Neighbor Attention

研究动机

节点的邻居在消息传递过程中传递的信息量不一致。信息的传递应由节点之间的相关性决定：相似度越高，传递的信息越多。

虽然 Neighbor Normalization 已经对邻居借点进行了区分，但是这些区分仅考虑了节点的个体特征，而没有考虑节点在聚合过程中的重要性。

Neighbor Attention 的基本模式：

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v$$



Neighbor Attention

常见的形式：GAT

$$\alpha_{u,v} = \frac{\exp(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v])}{\sum_{v' \in \mathcal{N}(u)} \exp(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}])}$$

或者 [Bahdanau et al., 2014]

$$\alpha_{u,v} = \frac{\exp(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_v)}{\sum_{v' \in \mathcal{N}(u)} \exp(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_{v'})}$$

$$\alpha_{u,v} = \frac{\exp(\text{MLP}(\mathbf{h}_u, \mathbf{h}_v))}{\sum_{v' \in \mathcal{N}(u)} \exp(\text{MLP}(\mathbf{h}_u, \mathbf{h}_{v'}))}$$



Neighborhood Aggregation 总结

常见方法

- ① Neighbor Normalization
- ② Neighbor Pooling
- ③ Neighbor Attention

Aggregation 的核心是 permutation invariant.

挑战

- ① Permutation Invariant 是否必须要满足?
- ② 如何建模邻居之间的差异?
- ③ 复杂网络 (如异构图) 如何建模?

Update

定义

Neural Message Passing 中的 Update function 目标是把节点自身的信息和邻居信息整合，并更新节点自身的表征。

$$h'_u = f_{update}(h_u, h_{\mathcal{N}(u)})$$

简易的 Update function 包括 concatenation 和求和等。更为复杂的 Update 模块设计则着重关注 Over-smoothing 问题。

Over-smoothing Problem

过度平滑 (Over-Smoothing Problem) 是指图神经网络在更新的过程中，由于会搜集多层邻居的信息，经过多次更新之后，导致所有节点的表征都非常相似，难以区分。

Over-smoothing 问题理论分析

网络中的两个节点在表征学习结果中的影响可以用 Jacobian Matrix 表示：

$$I_K(u, v) = \mathbf{1}^\top \left(\frac{\partial \mathbf{h}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} \right) \mathbf{1}$$

$$I_K(u, v) \propto p_{\mathcal{G}, K}(u \mid v)$$

节点 u 对节点 v 的影响正比于从节点 u 出发的长度为 K 的随机游走到节点 v 的概率。当 K 趋向于无穷大时等价于稳态分布，此时与邻居无关，只与节点自身有关。

对于存在度很大的节点的图，整个图很容易就走到稳态



Concatenation and Skip-connection

Over-smoothing 的信息保留

随着搜集的信息越来越多，节点自身的特性逐渐消失，update 之后的信息大多来自于邻居节点。Update 过程希望能够区分来自节点自身的信息还是其邻居节点传递的信息。

$$\text{UPDATE}_{\text{concat}} (\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = [\text{UPDATE}_{\text{base}} (\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) \oplus \mathbf{h}_u]$$

常见的方法包括 vector concatenation 和 skip connection



研究动机

受到 GRU (Gated Recurrent Unit) 模块在 RNN 模型中的显著效果启发，使用 GRU 模块更新节点表示：

$$\mathbf{h}_u^{(k)} = \text{GRU} \left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} \right)$$

理论上，所有用于更新 RNN 的状态的模块都可以用于更新 GNN
LSTM GRU 是目前防止 oversmoothing 最好的手段之一。

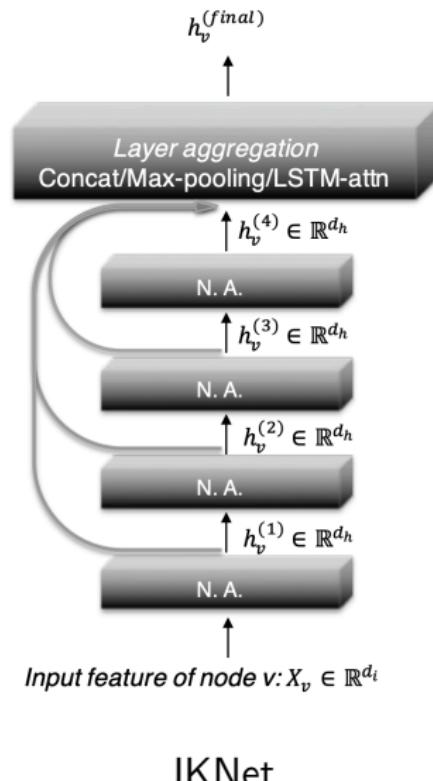


Jumping Knowledge Connections⁶

研究动机

为了 Jumping Knowledge Connection 选取了所有的层的表示融合得到最终的表示，从而防止太深的层带来的噪声信息以及过度平滑。

$$\mathbf{z}_u = f_{JK} \left(\mathbf{h}_u^{(0)} \oplus \mathbf{h}_u^{(1)} \oplus \dots \oplus \mathbf{h}_u^{(K)} \right)$$



JKNet

⁶Representation Learning on Graphs with Jumping Knowledge

通用 GNN 框架

GNN 模型的通用框架可以表示为：

$$\mathbf{h}_{(u,v)}^{(k)} = \text{UPDATE}_{\text{edge}} \left(\mathbf{h}_{(u,v)}^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{G}}^{(k-1)} \right)$$

$$\mathbf{m}_{\mathcal{N}(u)} = \text{AGGREGATE}_{\text{node}} \left(\left\{ \mathbf{h}_{(u,v)}^{(k)} \forall v \in \mathcal{N}(u) \right\} \right)$$

$$\mathbf{h}_u^{(k)} = \text{UPDATE}_{\text{node}} \left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}, \mathbf{h}_{\mathcal{G}}^{(k-1)} \right)$$

$$\mathbf{h}_{\mathcal{G}}^{(k)} = \text{UPDATE}_{\text{graph}} \left(\mathbf{h}_{\mathcal{G}}^{(k-1)}, \left\{ \mathbf{h}_u^{(k)} \forall u \in \mathcal{V} \right\}, \left\{ \mathbf{h}_{(u,v)}^{(k)} \forall (u,v) \in \mathcal{E} \right\} \right)$$



GNN 模型训练

GNN 模型的训练方式大致可以分为两大类：半监督，无监督

半监督 GNN

半监督 GNN 的最大化

$$\mathcal{L} = \sum_{u \in \mathcal{V}_{\text{train}}} -\log(\text{softmax}(\mathbf{z}_u, \mathbf{y}_u))$$

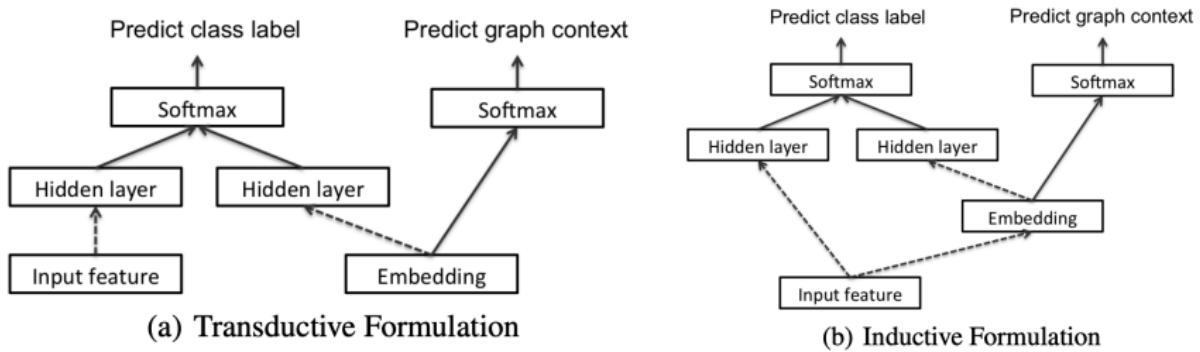
$$\text{softmax}(\mathbf{z}_u, \mathbf{y}_u) = \sum_{i=1}^c \mathbf{y}_u[i] \frac{e^{\mathbf{z}_u^\top \mathbf{w}_i}}{\sum_{j=1}^c e^{\mathbf{z}_u^\top \mathbf{w}_j}}$$

无监督 GNN

无监督 GNN 的目标是使用节点的表征拟合节点的临近性

$$J_{\mathcal{G}}(z_u) = -\log(\sigma(z_u^T z_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-z_u^T z_{v_n}))$$

Transductive VS Inductive



Transductive 任务 VS Inductive 任务⁷



⁷Revisiting Semi-Supervised Learning with Graph Embeddings(ICML 2016)

总结

- ① 基于深度自编码器的节点表征学习
- ② 融合节点属性的节点表征学习
- ③ 基于神经消息传递的节点表征学习



References I

-  Bahdanau, D., Cho, K., and Bengio, Y. (2014).
 Neural machine translation by jointly learning to align and translate.
arXiv preprint arXiv:1409.0473.
-  Wang, D., Cui, P., and Zhu, W. (2016).
 Structural deep network embedding.
 In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234.
-  Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. (2017).
 Deep sets.
arXiv preprint arXiv:1703.06114.

