

# 图神经网络导论

## 初探图神经网络

授课教师：周晟

浙江大学 软件学院

2023.11



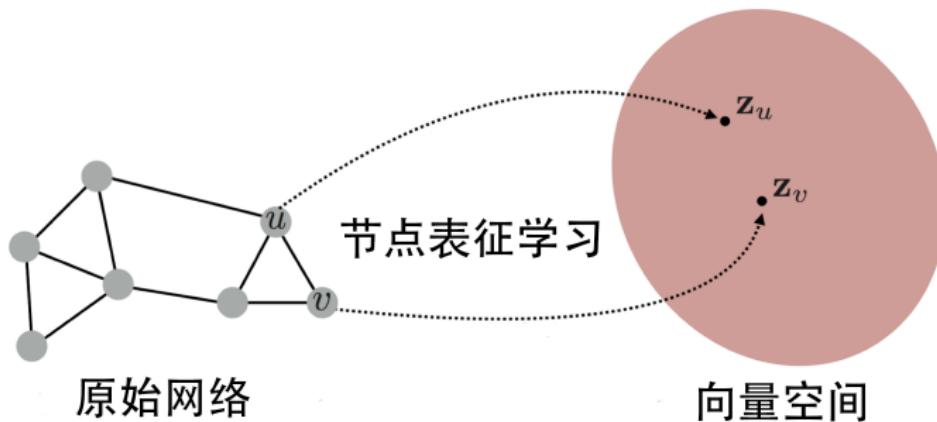
# 上节回顾

## 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



# 上节回顾

经典图表征学习算法与矩阵分解的等价性<sup>1</sup>:

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	$\log \left( \frac{\frac{1}{2T} \sum_{r=1}^T (\sum_u \mathbf{X}_{w,u} \mathbf{P}_{c,w,u}^r + \sum_u \mathbf{X}_{c,u} \mathbf{P}_{w,c,u}^r)}{(\sum_u \mathbf{X}_{w,u})(\sum_u \mathbf{X}_{c,u})} \right) - \log b$

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  $D_{\text{col}}$  &  $D_{\text{row}}$ .

$D = \text{diag}(d_1, \dots, d_{|\mathcal{V}|})$ , 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.

<sup>1</sup>Network 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<sup>2</sup>:

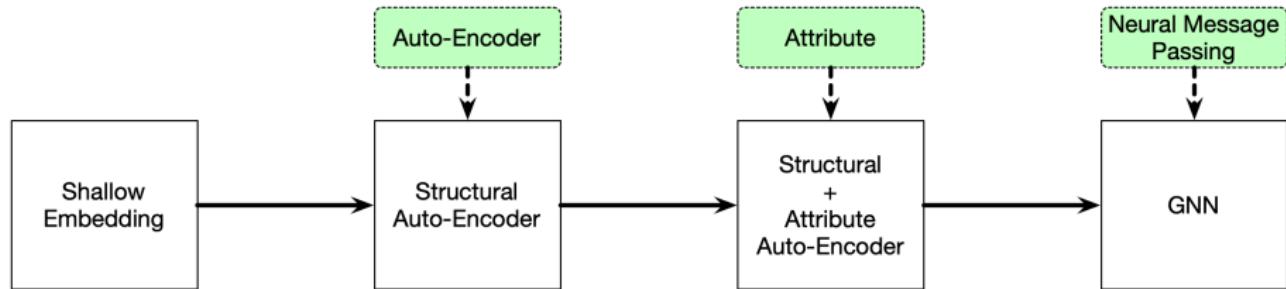
- ① 仅用于拟合节点的临近信息，无法拟合其他结构信息
- ② 无法融入属性等其他信息
- ③ 难以支持批量训练

<sup>2</sup>Representation 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

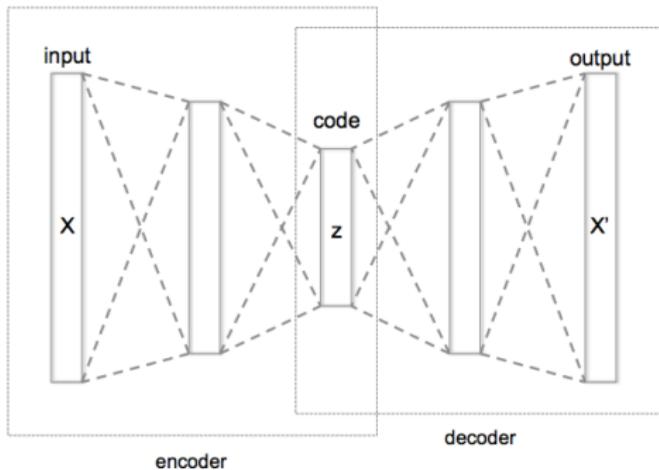
④ 经典图神经网络



# 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 ...

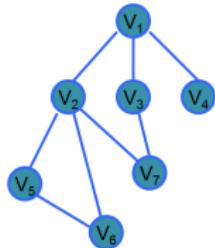
深度自编码器在计算机视觉，自然语言处理等领域已取得广泛的应用和成功。

使用深度自编码器提取网络中节点的特征？

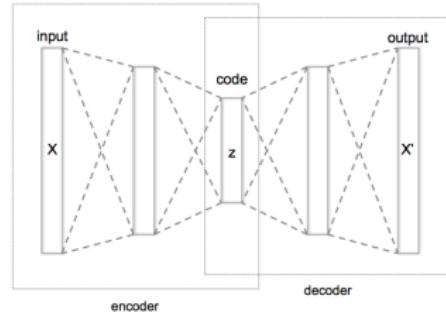


# 基于深度自编码器的节点表征学习

一种简单的基于深度自编码器的节点表征学习模型：

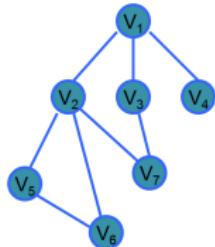


	0	1	2	3	4	5	6
0	0	1	1	1	0	0	0
1	1	0	0	0	1	1	1
2	1	0	0	0	0	0	1
3	1	0	0	0	0	0	0
4	0	1	0	0	0	1	0
5	0	1	0	0	1	0	0
6	0	1	1	0	0	0	0

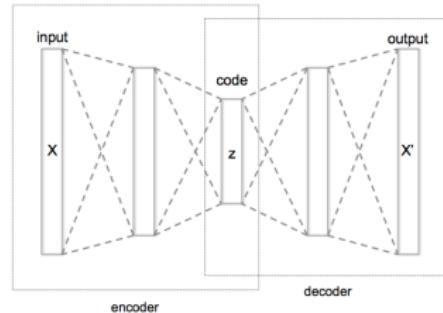


# 基于深度自编码器的节点表征学习

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

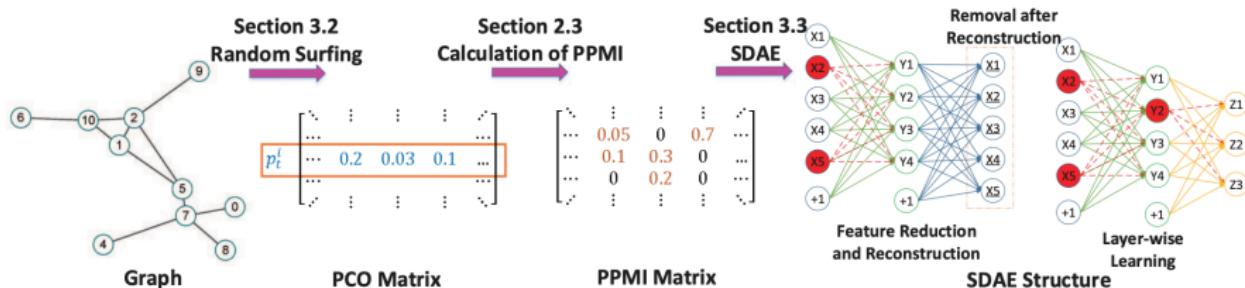


## 缺陷

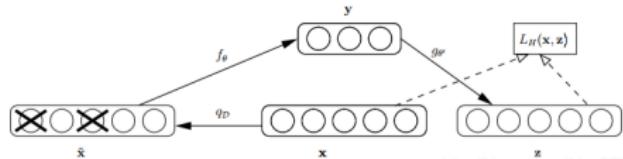
- ① 模型容易坍塌
- ② 无法捕获高阶结构特征
- ③ ...



# 被编码数据的改进



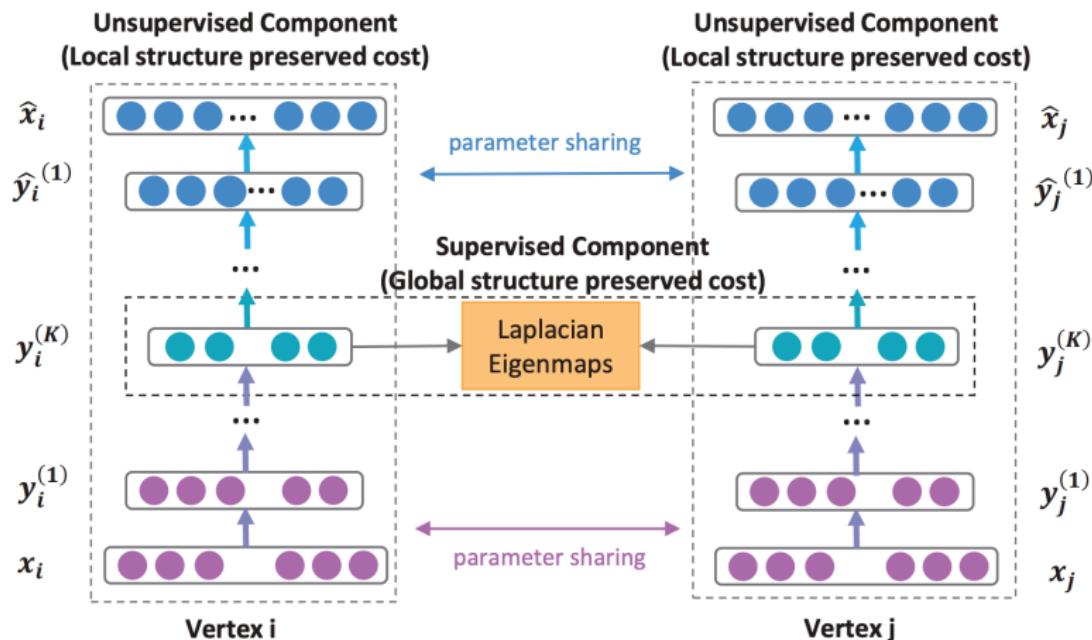
SDAE<sup>3</sup>: 用 PPMI 矩阵代替原始邻接矩阵



Denoised Auto-Encoder

<sup>3</sup>Deep 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]

## 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<sup>4</sup>

---

<sup>4</sup>Network 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

④ 经典图神经网络



# 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**:

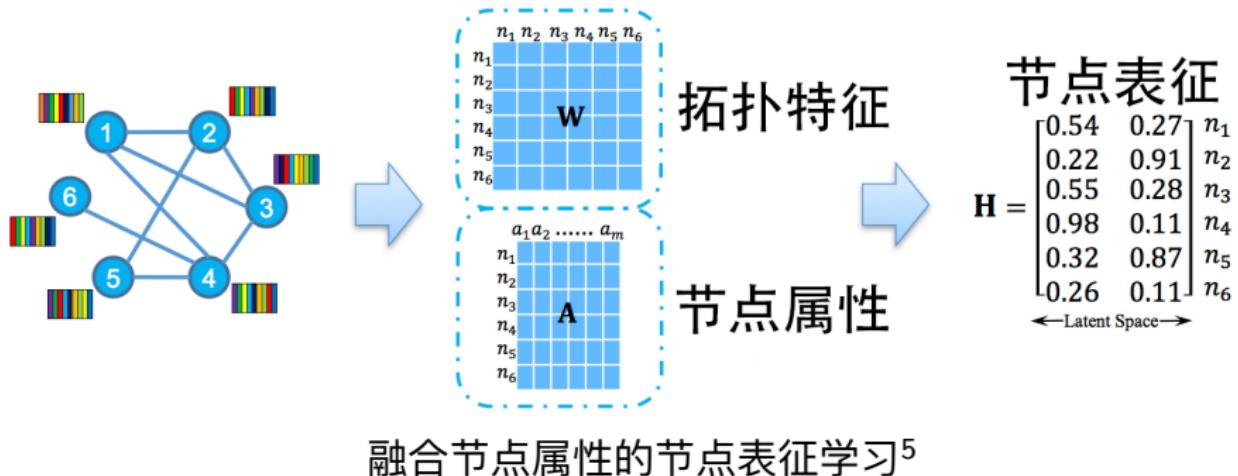
- ① 缺失信息补充
- ② 结构信息降噪
- ③ 克服稀疏性问题
- ④ 刻画图数据特点

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

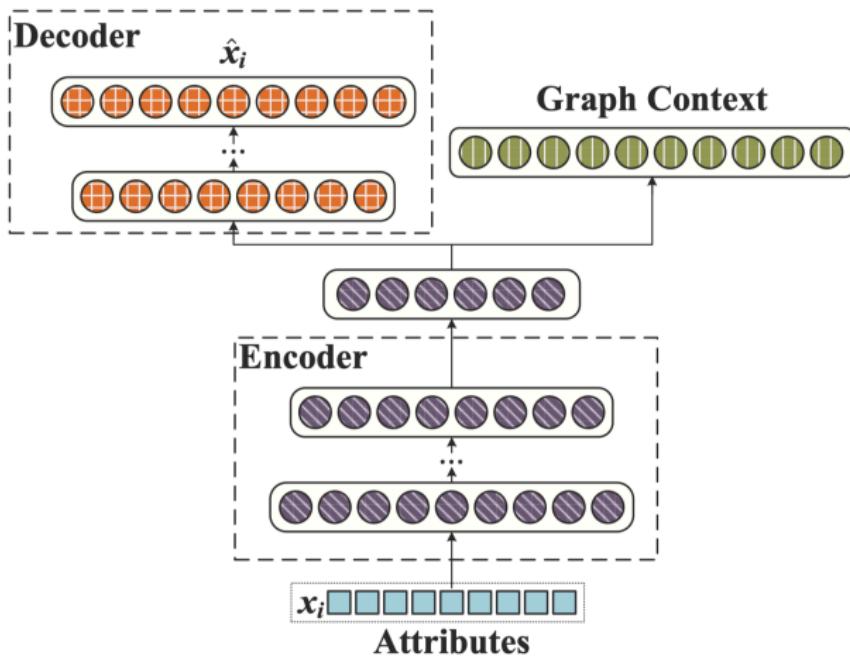


$$\mathcal{L} = \|\mathbf{S} - \mathbf{H}\mathbf{H}^T\|_F^2 + \lambda \sum_{(i,j) \in \mathcal{E}} w_{ij} \|h_i - h_j\|_2$$



<sup>5</sup> 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

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

$$p(v_{i+j} | \mathbf{x}_i) = \frac{\exp(\mathbf{v}'^T f(\mathbf{x}_i))}{\sum_{v=1}^n \exp(\mathbf{v}'^T f(\mathbf{x}_i))}$$

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	<b>0.729</b>	<b>0.673</b>	<b>0.876</b>	<b>0.871</b>	<b>0.759</b>	<b>0.755</b>

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。

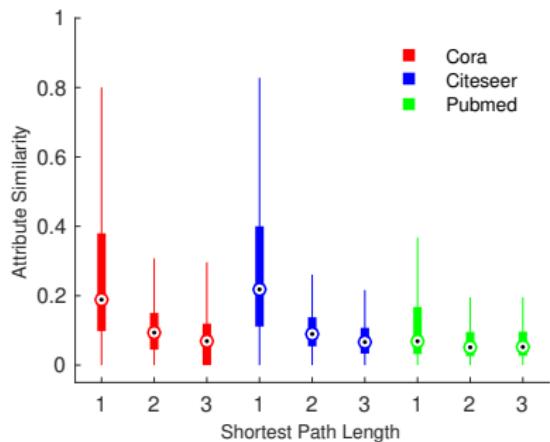
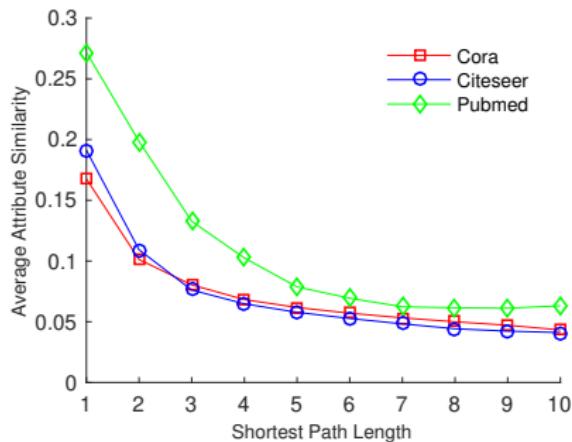
现象：

- 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

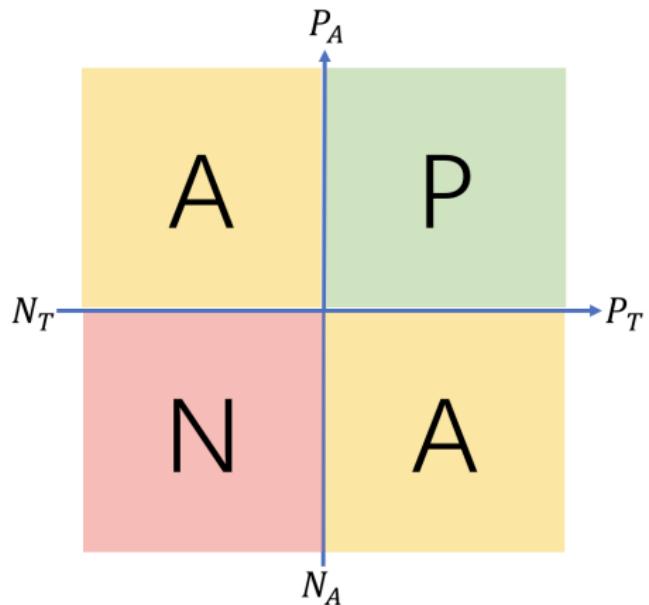


# PRRE: Personalized Relation Ranking Embedding for Attributed Networks

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  $\theta$  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：

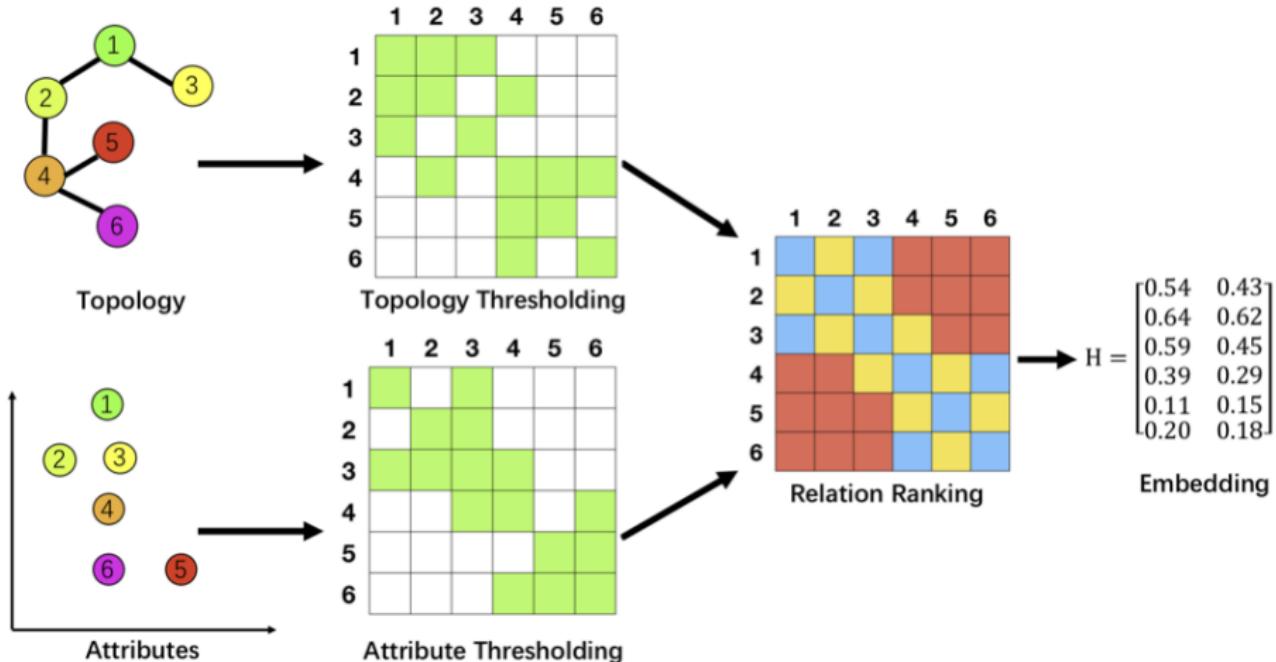
$$\prod_{\{i,j,k\} \in \mathcal{V}} P(j \geq_i k | \theta, H) = \left[ \frac{\sigma_{ij} - \sigma_{ik} + 1}{2} \right]^{\frac{1}{1+g(\theta)}},$$

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

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

④ 经典图神经网络



## 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 同质性假设
- ② 排列不变性

# 同质性假设

## 同质性 (Homophily) 假设

同质性 (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

节点的同质性：

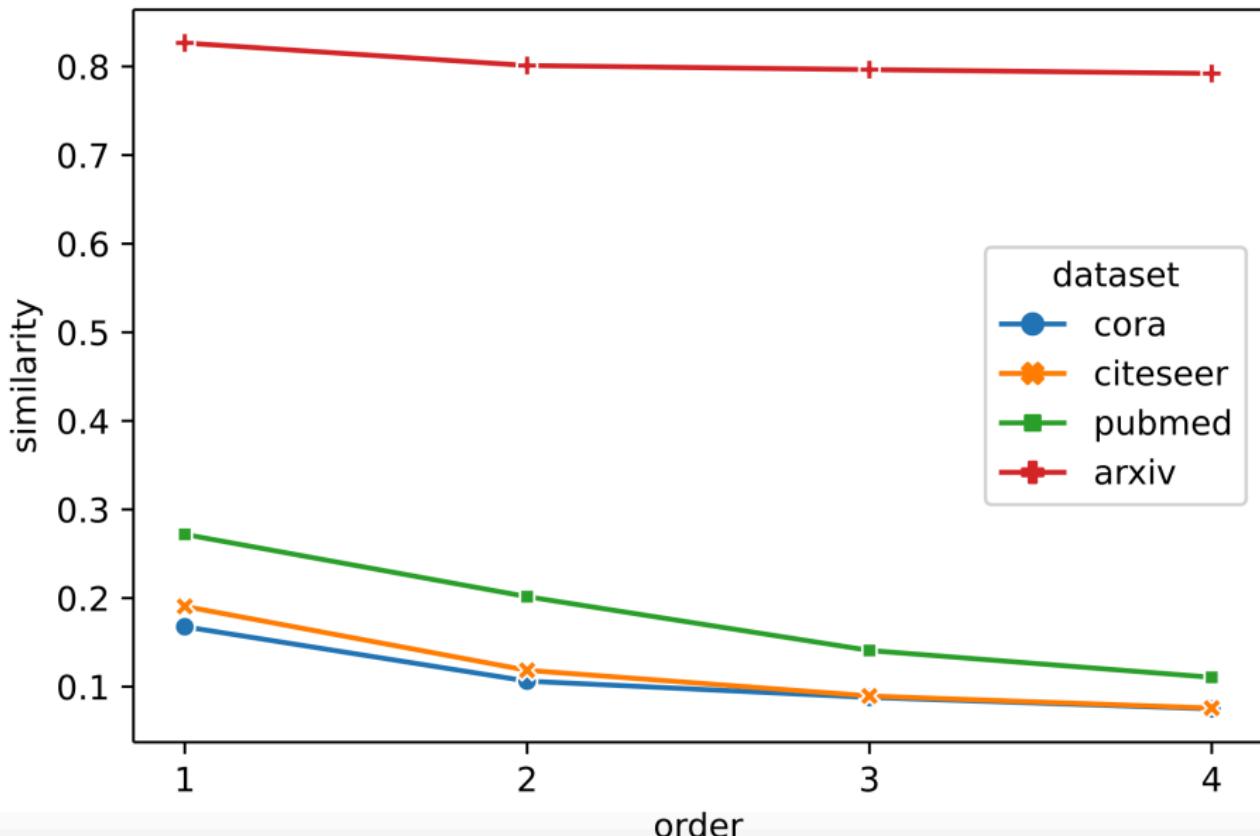
$$\mathcal{H}_{\text{node}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{u \in \mathcal{N}(v) : y_v = y_u\}|}{|\mathcal{N}(v)|}$$

边的同质性：

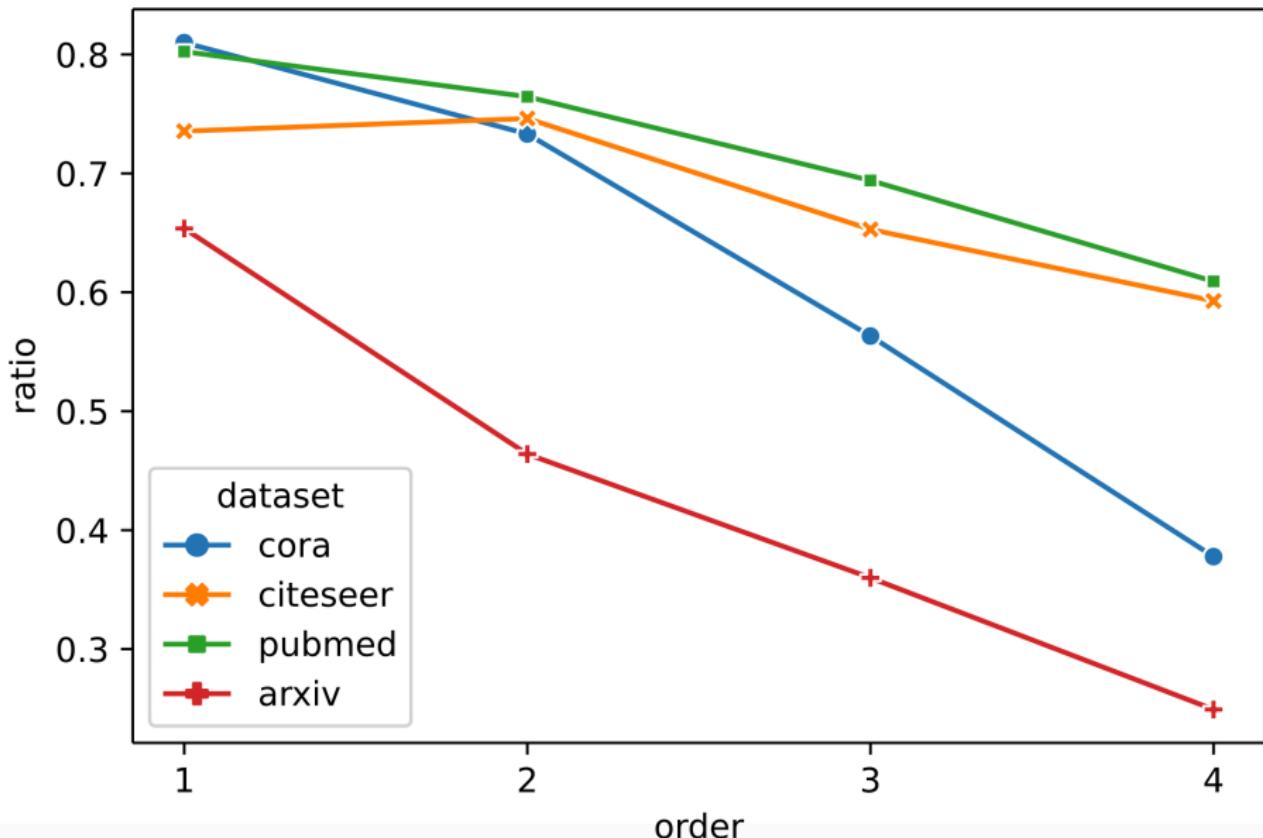
$$\mathcal{H}_{\text{edge}} = \frac{|\{(v, u) \in \mathcal{E} : y_v = y_u\}|}{|\mathcal{E}|}.$$



# 拓扑相似性与属性相似性



# 拓扑相似性与标签一致性



# 同质性假设

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

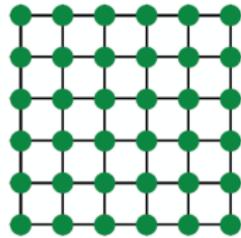
# 异质性 (Heterophily) 网络

# 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。

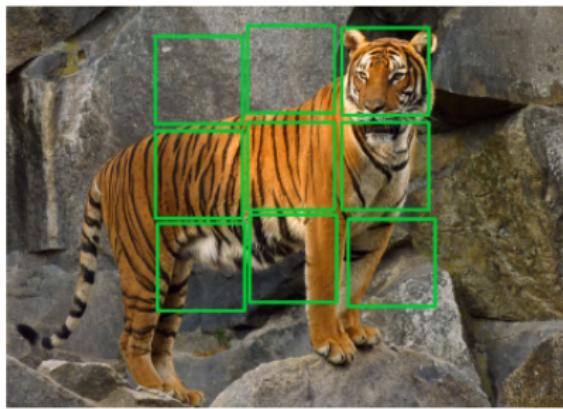


# 图数据的排列不变性

## 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)。

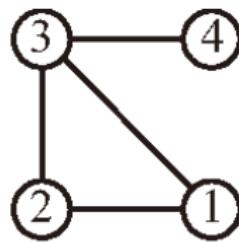
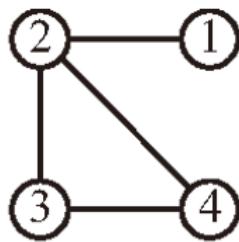
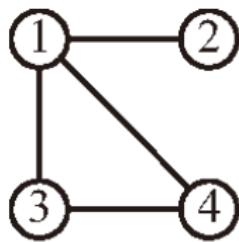
$$f(\mathbf{PAP}^\top) = f(\mathbf{A})$$
$$f(\mathbf{PAP}^\top) = \mathbf{P}f(\mathbf{A})$$



# 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。



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

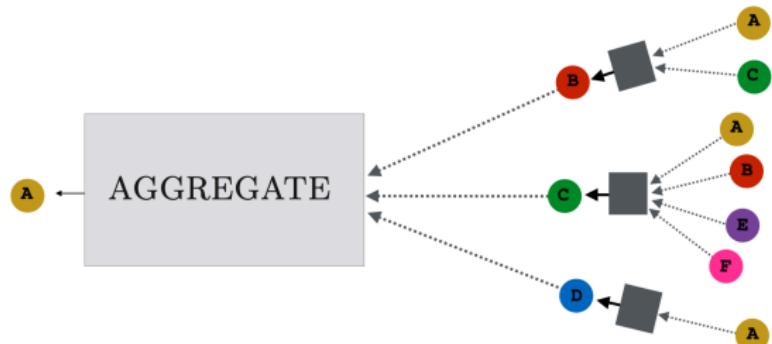
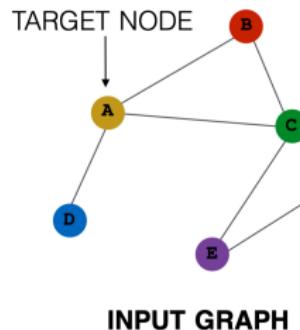
$$\begin{bmatrix} 0, 1, 0, 0 \\ 1, 0, 1, 1 \\ 0, 1, 0, 1 \\ 0, 1, 1, 0 \end{bmatrix}$$

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

# 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

同质性假设的存在，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:

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

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

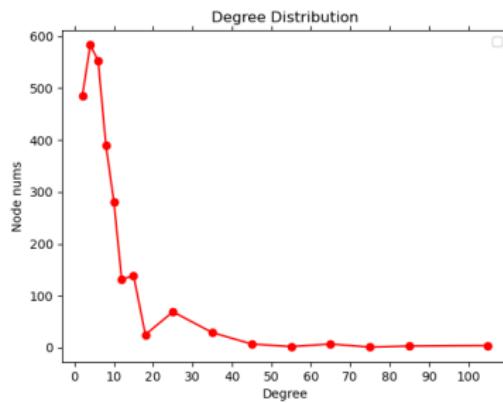
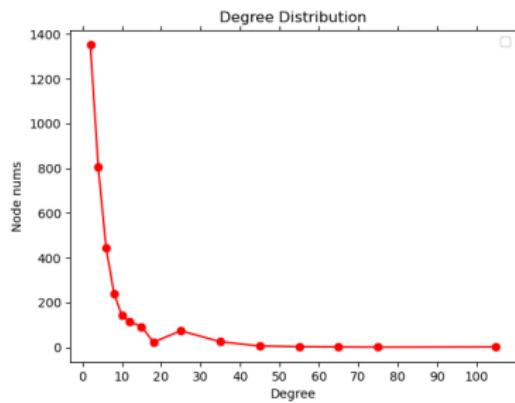
## 缺陷

- 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。

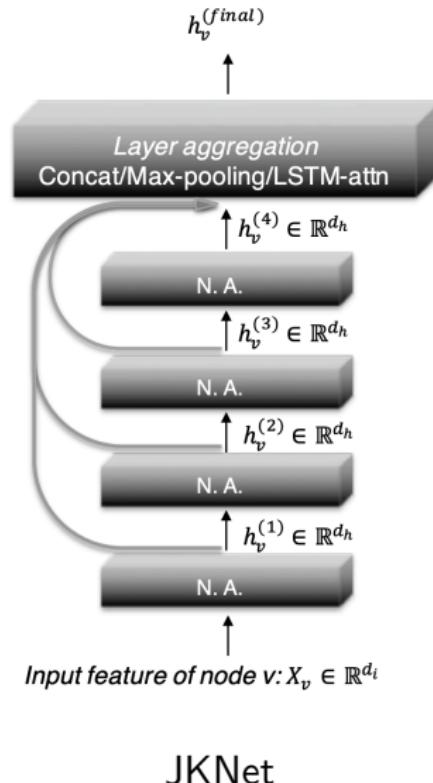


# Jumping Knowledge Connections<sup>6</sup>

## 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

<sup>6</sup>Representation 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

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

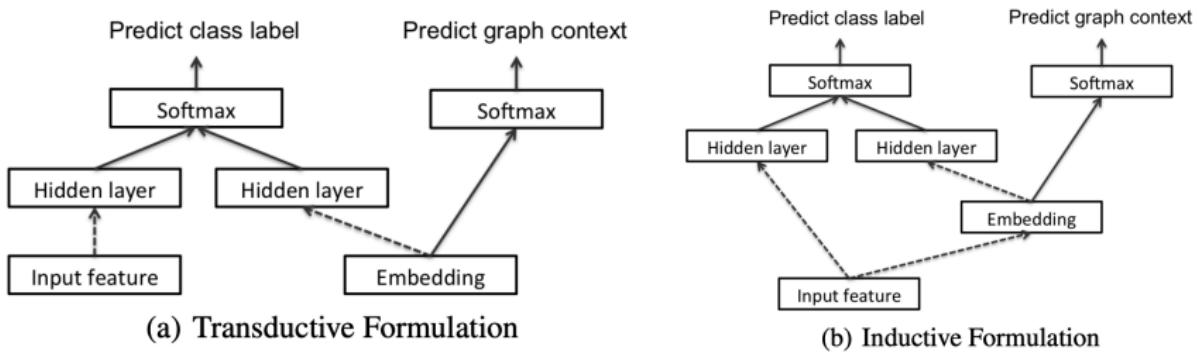
$$\text{softmax}(\mathbf{z}_u, \mathbf{y}_u) = \sum_{i=1}^c \mathbf{y}_u[i] \frac{e^{\mathbf{z}_u^\top \mathbf{w}_i}}{\sum_{j=1}^c e^{\mathbf{z}_u^\top \mathbf{w}_j}}$$

## 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<sup>7</sup>



<sup>7</sup>Revisiting Semi-Supervised Learning with Graph Embeddings(ICML 2016)

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

④ 经典图神经网络

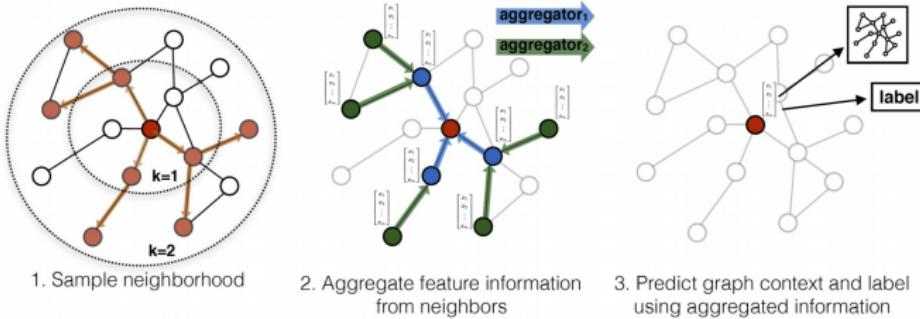


# GraphSAGE[Hamilton et al., 2017]

## 研究动机

现有的 Graph Embedding 的方法大多要求图中所有的点在训练阶段出现过，对于训练阶段未出现的节点学习节点表征，需要对整个网络进行重新训练。

## Transductive VS Inductive



GraphSAGE 由邻居采样、邻居特征聚集和训练三部分组成

# GraphSAGE——邻居节点采样

## 研究动机

- ① 网络中存在度比较大的节点，整合邻居信息复杂度高
- ② 网络中节点度差异较大，模型训练不稳定

## GraphSAGE 邻居节点采样

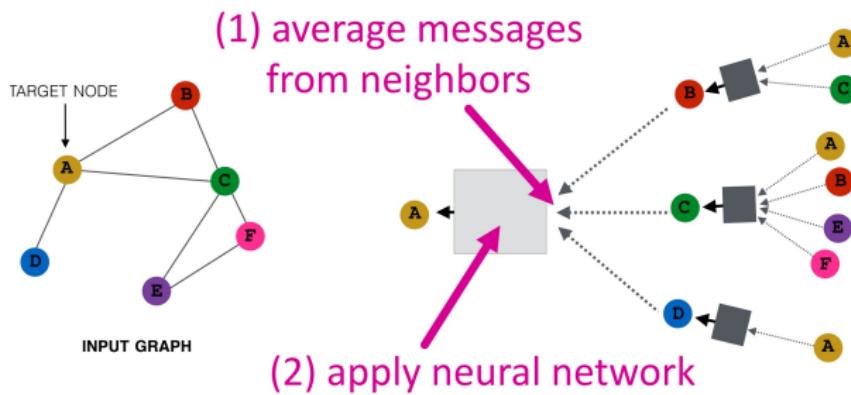
对于每个节点采样固定数量  $S$  的邻居，若节点邻居数小于  $S$ ，则采用有放回的抽样方法，否则采用无放回的抽样。

网络的不同层之间使用独立的 Random Walk 采样，同一个节点在不同层采样得到邻居的概率是不同的。

# GraphSAGE——Aggregation

在 GraphSAGE 的每一层，都执行两个操作：

- ① 从邻居节点中聚集特征
- ② 通过神经网络生成目标节点的特征



# GraphSAGE——Aggregation

---

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

---

**Input :** Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ ; input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$ ; depth  $K$ ; weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$ ; non-linearity  $\sigma$ ; differentiable aggregator functions  $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$ ; neighborhood function  $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations  $\mathbf{z}_v$  for all  $v \in \mathcal{V}$

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;  
2 for  $k = 1 \dots K$  do  
3   for  $v \in \mathcal{V}$  do  
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;  
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$   
6   end  
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$   
8 end  
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

---

GraphSAGE 的前向传播

# GraphSAGE——Aggregation

GraphSAGE aggregation 函数：

$$h_v^k = \sigma \left( W^k \cdot \text{CONCAT} \left( h_v^{k-1}, \text{AGG} \left( \{h_u^{k-1} \mid \forall u \in \mathcal{N}(v)\} \right) \right) \right)$$

GraphSAGE 在聚合过程中，对于每一层的节点表示都进行了 L2 的标准化。

$$h_v^k \leftarrow \frac{h_v^k}{\|h_v^k\|_2}, \forall v \in \mathcal{V} \text{ where } \|u\|_2 = \sqrt{\sum_i u_i^2}$$

表征的标准化使得所有的向量都有了相同的尺度。一定程度上可以使模型训练更稳定，并带来一定的效果提升。

# GraphSAGE——Aggregation

为了在 Aggregation 过程中保留图的 permutation invariant,  
GraphSAGE 中尝试了三种聚集函数：

- Mean aggregator
- LSTM aggregator
- Pooling aggregator



# GraphSAGE——Mean AGG

$$h_v^k \leftarrow \sigma(W_k \cdot \text{MEAN}(\{h_v^{k-1}\} \cup \{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}))$$

- 在 Mean aggregator 中， MEAN 操作替代了 CONCAT 操作。除了对邻居节点的特征求平均，还将目标节点本身也看作了它的邻居节点，权重是相同的。
- 与 CONCAT 的区别： CONCAT 对于一个目标节点输出的 size 是 (1, 2D)，再通过  $W_k$  来变换成 (1, D)。而 Mean aggregator 的结果 size 是 (1, D)。



# GraphSAGE——LSTM AGG

- 相较于 Mean aggregator, LSTM aggregator 具有更强的表示能力，但它本质上并不是对称的，需要按照输入的顺序来处理数据。
- 因此，作者将邻居节点的向量集合随机打乱顺序，作为 LSTM aggregator 的输入。

$$AGGREGATE_k^{LSTM} = \text{LSTM}([h_u^{k-1}, \forall u \in \pi(\mathcal{N}(v))])$$

其中  $\pi(\cdot)$  表示随机打乱。



# GraphSAGE——Pooling AGG

$$AGGREGATE_k^{pool} = \max(\{\sigma(W_{pool}h_u^{k-1} + b_{pool}), \forall u \in \mathcal{N}(v)\})$$

- 每个邻居节点都通过一个全连接网络，然后整体做 max-pooling。权重  $W_{pool}$  和  $b_{pool}$  是共享的。
- 任何满足对称性的算子都可以用来替换 max，比如 mean。在实验中，作者发现 mean-pooling 和 max-pooling 并没有显著的差别。



# GraphSAGE——训练

## 无监督训练

节点的低维表征可以拟合节点之间的相似性：

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

## 半监督训练

节点的低维表征可以最大化半监督数据的似然：

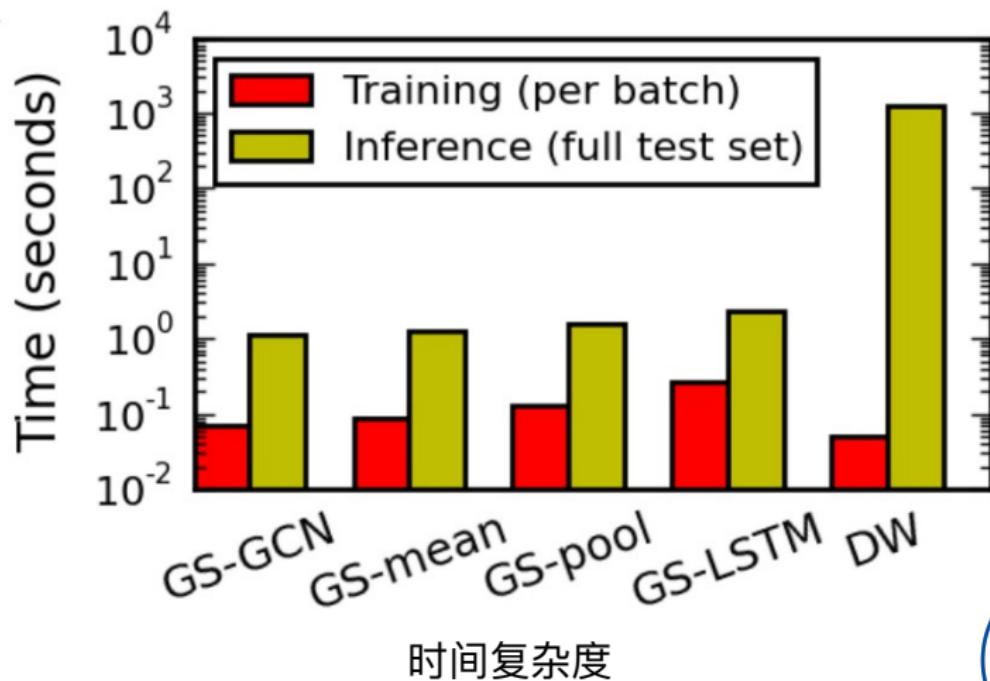
$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf}$$

# GraphSAGE——数据集

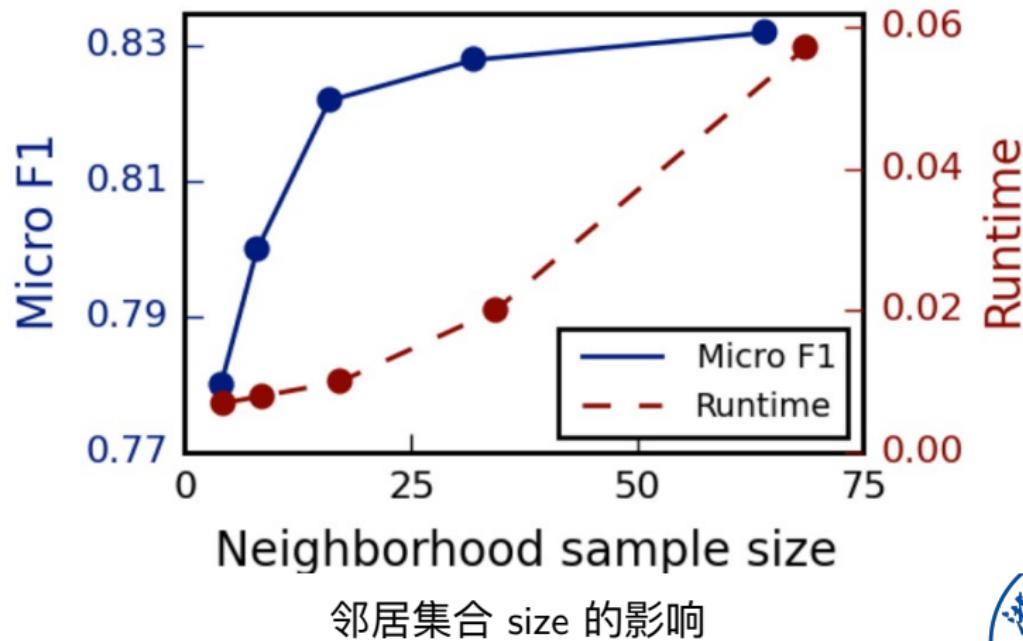
- Citation: 论文分类任务 (节点分类)  
包含 2000 年到 2005 年六个生物学相关领域的论文，每篇论文属于六种类别之一。数据集包含 302424 个节点，平均度数为 9.15。
- Reddit: 帖子分类任务 (节点分类)  
包含 Reddit 上发布帖子的一个大型图数据集，节点标签为帖子所属的社区。数据集包含 232965 个节点，平均度数为 492。
- PPI: 蛋白质分类任务 (Graph 分类)  
每个图对应一个不同的人体组织，共有 121 种标签。  
平均每个图包含 2373 个节点，平均度数为 28.8。



# GraphSAGE——实验结果



# GraphSAGE——实验结果



# GraphSAGE——实验结果

- 网络层数  $K$  (聚集  $K$  跳内的信息) 的设置：  
 $K=2$  相较  $K=1$  有  $10 \sim 15\%$  的效果提升；若  $K>2$ ，效果上只有  $0 \sim 5\%$  的提升，但计算时间却变大了  $10 \sim 100$  倍。因此  $K$  设置为 2。
- LSTM 和 pooling 的效果较好，但 LSTM 耗时比 pooling 大很多，所以 pooling 在总体上略占优势。



# Graph Attention Network[Veličković et al., 2017]

GraphSAGE-mean 也可以写为

$$h_v^k = \sigma \left( \sum_{u \in \mathcal{N}(v)} \alpha_{vu} W^k h_u^{k-1} \right)$$

其中  $\alpha_{vu} = \frac{1}{|\mathcal{N}(v)|}$  为节点  $u$  对于  $v$  的权重因子（重要性）。

在 GraphSAGE 中，所有邻居节点  $u \in \mathcal{N}(v)$  对于节点  $v$  的重要性都是相同的。

节点从相似的邻居搜集更多的信息，从不相似的邻居搜集更少的信息

# GAT——attention

## Attention 机制

Attention 机制通过在模型训练过程中动态调整模型对数据/特征不同部分的权重提升模型的学习效果。在计算机视觉，自然语言处理中取得了广泛的应用。

input image



attention map



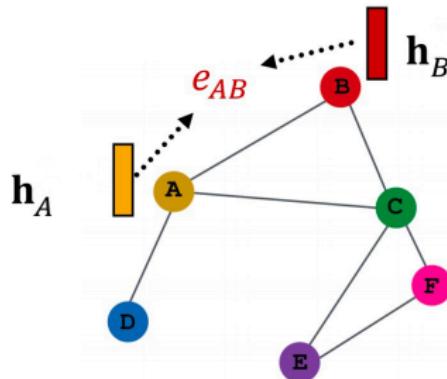
GAT 在图神经网络中引入了 Attention，希望关注邻居节点集合中比较重要的部分，淡化其余部分。

# GAT——注意力机制

- GAT 对节点执行一个共享的注意力机制  $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ ,  
由此计算 attention 系数:

$$e_{ij} = a(W\vec{h}_i, W\vec{h}_j)$$

表示节点  $j$  的特征对节点  $i$  的重要性。



# GAT——注意力机制

- 在最一般的机制中，所有节点对之间的 attention 系数都会被计算，这种做法显然会丢失**结构信息**。
- GAT 采用了一种 masked attention 的方式——仅将注意力分配到节点  $i$  的邻居节点集  $\mathcal{N}_i$  上，并用 softmax 对 attention 系数进行了标准化：

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$



# GAT——注意力机制

- 在 GAT 中，注意力机制  $a$  是由一个单层前馈神经网络来实现的，其权重向量  $\vec{a} \in \mathbb{R}^{2F'}$ ，并使用 LeakyReLU 作为非线性激活函数。
- 由此，权重因子  $\alpha$  可以表示为：

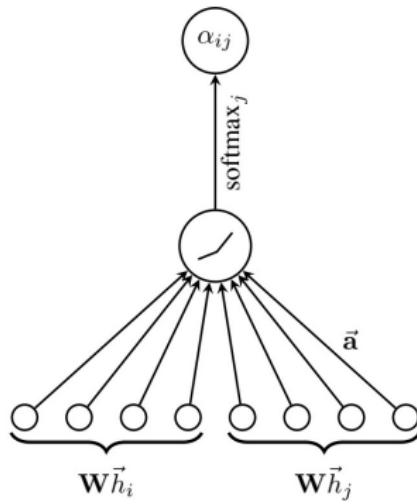
$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\vec{a}^T [W\vec{h}_i || W\vec{h}_j])))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\vec{a}^T [W\vec{h}_i || W\vec{h}_k]))}$$

其中  $||$  表示拼接操作。



# GAT——注意力机制

$$\alpha_{ij} = \frac{\exp(\text{LeakReLU}(\vec{a}^T [\vec{W}\vec{h}_i || \vec{W}\vec{h}_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\vec{a}^T [\vec{W}\vec{h}_i || \vec{W}\vec{h}_k]))}$$

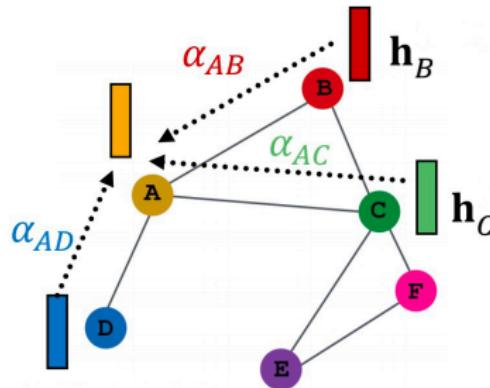


# GAT——注意力机制

- 网络的输出特征可以通过下式得到：

$$\vec{h}'_i = \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij} W \vec{h}_j \right)$$

$\sigma$  为一个非线性激活函数。



# GAT——multi-head attention

- 为了提高模型的拟合能力，GAT 还引入了 multi-head attention，即同时使用多个独立的注意力机制来计算 attention 因子，并将它们的结果合并（拼接或求和）：

$$\vec{h}'_i = \parallel_{k=1}^K \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j \right)$$

其中  $\parallel$  表示拼接操作， $\alpha_{ij}^k$  为第  $k$  个注意力机制  $a^k$  产生的 attention 因子。

由于  $W^k \in \mathbb{R}^{F' \times F}$ ，所以这里的  $\vec{h}'_i \in \mathbb{R}^{KF'}$ 。



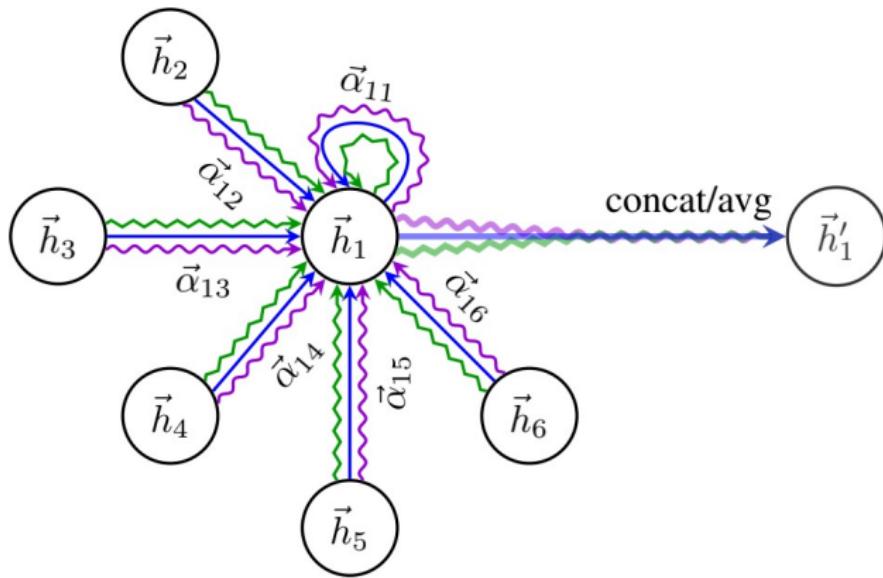
# GAT——multi-head attention

- 对于网络的最终层，拼接操作就不再合适了。因此，可以用求均值的方式来代替：

$$\vec{h}'_i = \sigma \left( \frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j \right)$$



- 下图为  $K=3$  的 multi-head attention 示意。不同颜色的线表示不同的 attention 机制。



# GAT——实验设置

- GAT 的实验建立在四个基于 Graph 的分类任务上，包括三个 transductive learning 任务和一个 inductive learning 任务。

	Cora	Citeseer	Pubmed	PPI
<b>Task</b>	Transductive	Transductive	Transductive	Inductive
<b># Nodes</b>	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)	56944 (24 graphs)
<b># Edges</b>	5429	4732	44338	818716
<b># Features/Node</b>	1433	3703	500	50
<b># Classes</b>	7	6	3	121 (multilabel)
<b># Training Nodes</b>	140	120	60	44906 (20 graphs)
<b># Validation Nodes</b>	500	500	500	6514 (2 graphs)
<b># Test Nodes</b>	1000	1000	1000	5524 (2 graphs)



# GAT——实验设置

- transductive learning:

在训练过程中已经用到了测试集数据（不带标签），只能预测其在训练过程中使用过的样本，对于新的节点需要重新进行训练。

- inductive learning:

训练过程中只使用训练集中的数据，预测时对于新的节点不需要重新训练。



# GAT——实验结果

Method	<i>Transductive</i>		
	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	<b>79.0%</b>
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	<b>79.0 ± 0.3%</b>
<b>GAT (ours)</b>	<b>83.0 ± 0.7%</b>	<b>72.5 ± 0.7%</b>	<b>79.0 ± 0.3%</b>

transductive learning 实验结果



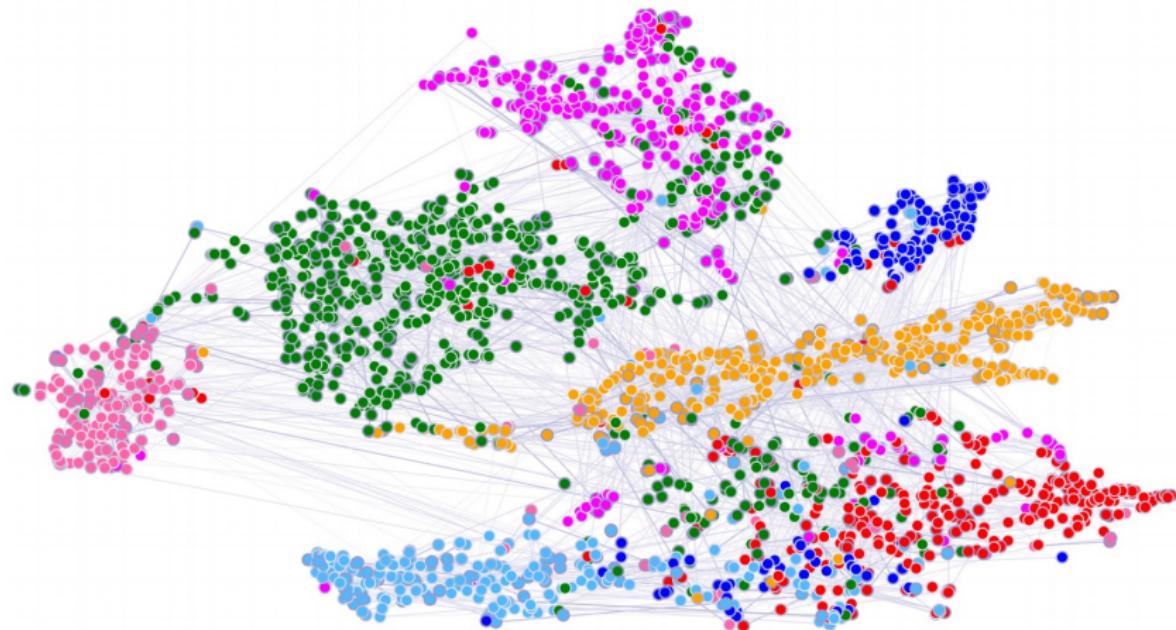
# GAT——实验结果

<i>Inductive</i>	
<b>Method</b>	<b>PPI</b>
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	$0.934 \pm 0.006$
<b>GAT</b> (ours)	<b><math>0.973 \pm 0.002</math></b>

inductive learning 实验结果



# GAT——t-SNE 可视化



GAT 在 Cora 数据集上使用 t-SNE 的可视化效果

# 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
- ④ 经典图神经网络



# 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.*
-  Hamilton, W. L., Ying, R., and Leskovec, J. (2017).  
 Inductive representation learning on large graphs.  
 In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.
-  Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017).  
 Graph attention networks.  
*arXiv preprint arXiv:1710.10903.*



# References II

- 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*.

