

Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心



图嵌入-GraphSAGE

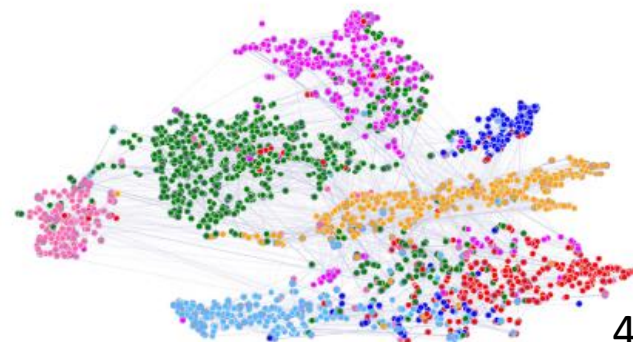
周妍汝 博士研究生

2020年07月05日

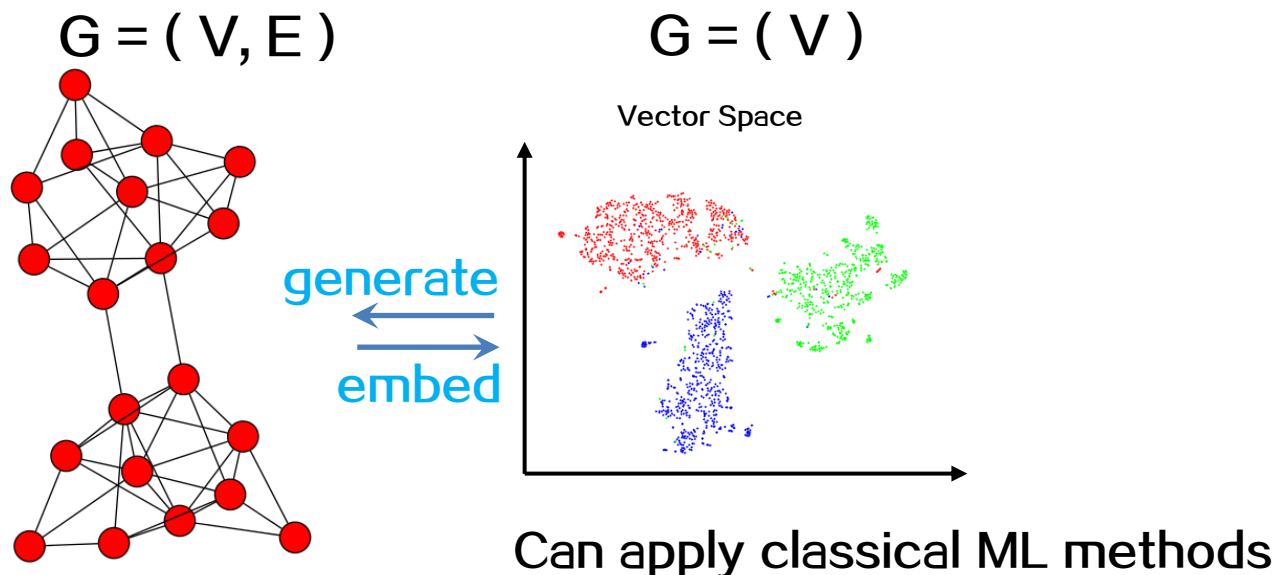
- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 应用总结
- 参考文献

- 预期收获
 - 1. 熟悉图嵌入的历史现状、分类及应用场景
 - 2. 理解Graphsage的算法原理
 - 3. 了解Graphsage的应用

- 图(Graph)是一个非常常用的数据结构，现实世界中很多很多任务可以描述为图问题
- Graph研究分类
 - 经典图算法：如生成树算法，最短路算法，二分图匹配，费用流问题等
 - 概率图模型：将条件概率表达为图结构，并进行进一步挖掘
 - 图神经网络：研究图结构数据挖掘问题



- 图嵌入 or 网络表示学习 or 网络嵌入
 - 旨在从图数据中学习得到图中每个节点的低维、实值、稠密的向量表示, 之后这些节点表示就可以作为节点的特征应用于后续的网络应用任务中, 如节点分类、链接预测、社区发现、可视化任务等。



- 有哪些图嵌入方法?
 - GraphSAGE(Graph SAmple and aggreGatE)

考虑网络结构

deepwalk

LINE

GraRep

node2vec

struc2vec

GraphSAGE

GraphWave

考虑结构和其他信息

CENE

CANE

Trans-Net

DL

GCN

SDNE

GAN

GraphGAN

ANE



基本概念

- 基本概念

- 直推学习 (Transductive learning)

- 训练集为 $D = \{X_{tr}, y_{tr}, X_{un}\}$, 测试集 X_{un} , 同时使用训练样本和测试样本来训练模型, 然后再次使用测试样本来测试模型效果。
 - “封闭世界” 不具备对未知数据的泛化能力。

- 归纳学习

- 训练集为 $D = \{X_{tr}, y_{tr}\}$, 测试集 (未标注) X_{te} , 从训练样本中学习规则然后应用在测试样本中。
 - “开放世界” 可以泛化到未知的样本

- 基本概念
 - 直推式图嵌入
 - 需要将整个图中的所有数据都加入到Embedding过程中，并且不能对新加入的数据及未知的数据进行Embedding。
 - 归纳式图嵌入
 - 不需要将整个图的数据全部加入到Embedding过程中，对未知的数据具有很好的泛化能力，在具有相同特征的图之间也有一定的泛化能力。



算法原理

T	基于节点特征（比如文本属性，节点信息，节点的度）学习生成未见过的节点的嵌入的函数
I	图 $G(V;E)$ ；所有节点的特征
P	1. 对图中每个节点邻居节点进行采样 2. 根据聚合函数聚合邻居节点蕴含的信息 3. 得到图中各节点的向量表示供下游任务使用
O	节点产生embedding的映射函数

P	如何归纳式学习实现未知节点embedding
C	边训练边归纳
D	如何从一个节点的局部邻居聚合特征信息
L	CCF A类会议

- **核心思想**
 - 去学习一个节点的信息是怎么通过其邻居节点的特征聚合而来的。当训练学习到了这样的“聚合函数”，就已知各个节点的特征和邻居关系，就可以很方便地得到一个新节点的表示
 - GraphSAGE方法学到的node embedding，是根据node的邻居关系的变化而变化的

- 生成节点embedding的前向传播算法

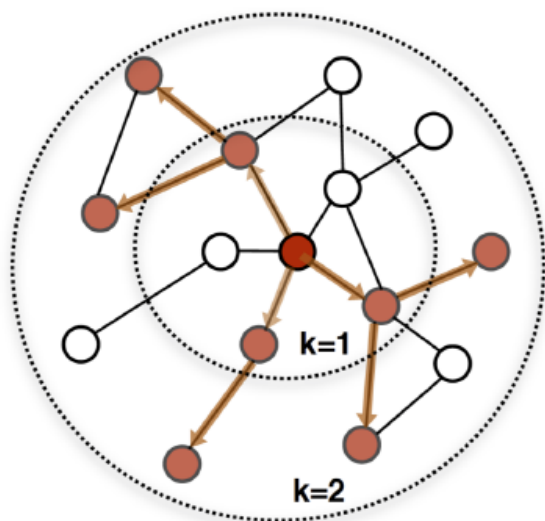
Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

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 σ ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

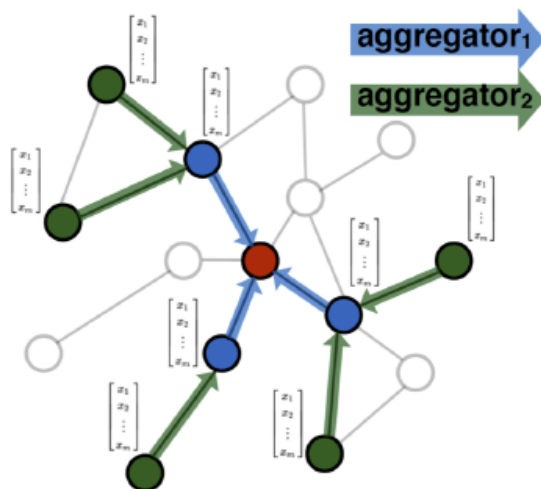
Output: Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

```
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      $\mathbf{h}_v^k \leftarrow \sigma \left( \mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k) \right)$ 
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  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

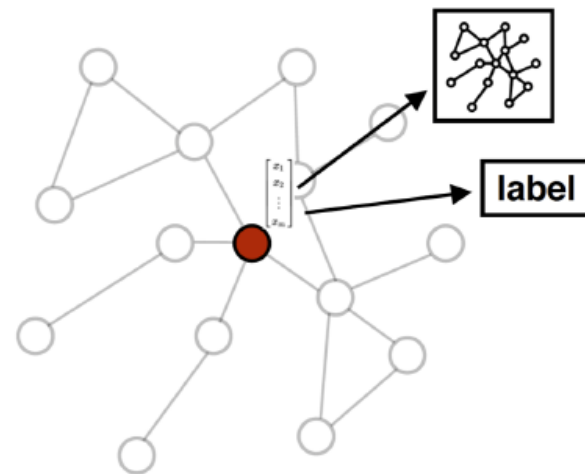
- 算法运行过程



1. Sample neighborhood

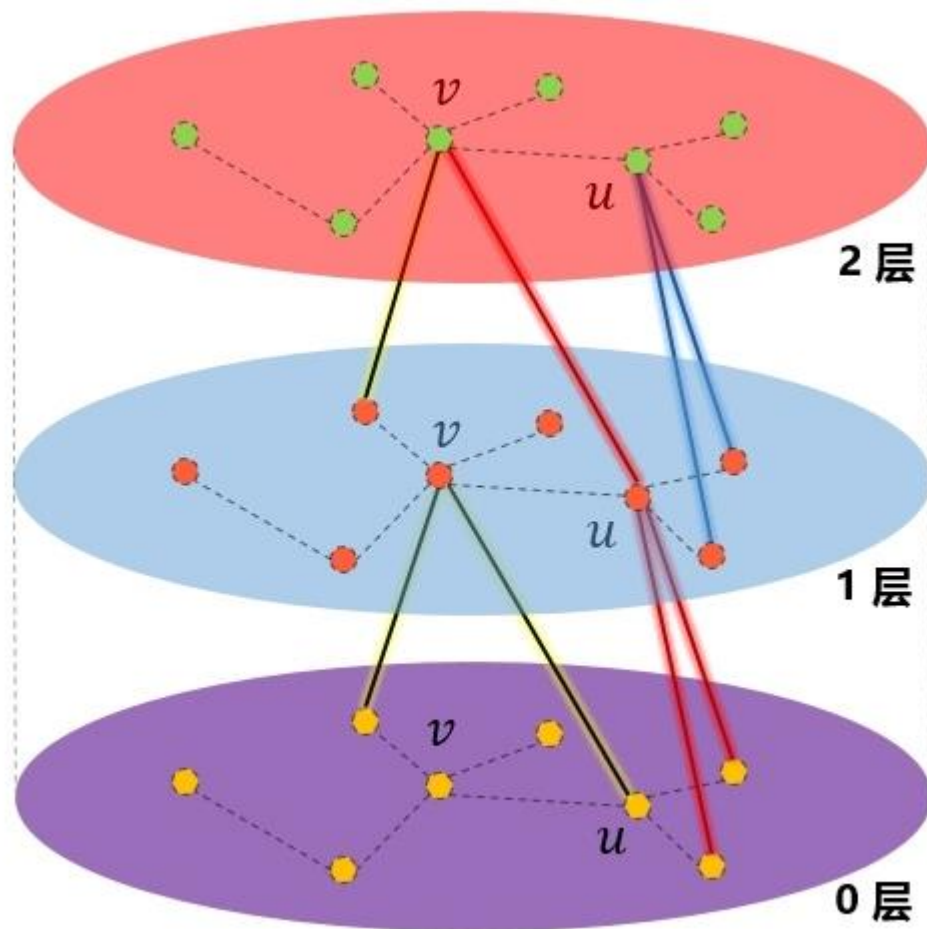


2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

- 采样邻居节点



- 聚合函数的选择
 - 需要满足排序不变量的特性，即输入顺序不会影响函数结果
 - 需要是对称的而且还要可训练
 - 4种聚合函数：平均、LSTM、pooling聚合器。

- 聚合函数的选择：
 - 平均聚合Mean aggregator
 - 先对邻居节点的第 $k-1$ 层向量中每个维度取平均，然后与目标节点向量进行拼接后，做一次非线性转换产生目标节点的第 k 层表示向量。

$$h_{N(v)}^k = \text{mean}(\{h_u^{k-1}, u \in N(v)\})$$
$$h_v^k = \sigma(W^k \cdot \text{CONCAT}(h_v^{k-1}, h_{N(v)}^k))$$

- 聚合函数的选择：
 - GCN归纳式聚合
 - 直接对目标节点和所有邻居embedding中每个维度取平均后再非线性转换：

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}))$$

- 原始伪代码：

$$\begin{aligned}\mathbf{h}_{\mathcal{N}(v)}^k &\leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}); \\ \mathbf{h}_v^k &\leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)\end{aligned}$$

- 与其他聚合函数区别：没有进行CONCAT操作

- 聚合函数的选择：
 - LSTM aggregator
 - 将LSTM应用在一个节点邻居的随机序列上，把所有邻居节点按随机排列输入LSTM，取最终隐状态为聚集之后的表示。

- 聚合函数的选择：
 - Pooling aggregator
 - 先对每个邻居节点k-1层向量输入到全连接神经网络中，然后进行一次max/mean pooling操作，将得到结果与目标顶点的表示向量拼接，最后再经过一次非线性变换得到目标顶点的第 k 层表示向量。

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma(\mathbf{W}_{\text{pool}} \mathbf{h}_{u_i}^k + \mathbf{b}), \forall u_i \in \mathcal{N}(v)\})$$

- 参数学习的损失函数
 - 无监督损失函数

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u^{\top} \mathbf{z}_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u^{\top} \mathbf{z}_{v_n}))$$

- 有监督损失函数

根据任务的不同直接设置目标函数即可，如最常用的节点分类任务使用交叉熵损失函数。

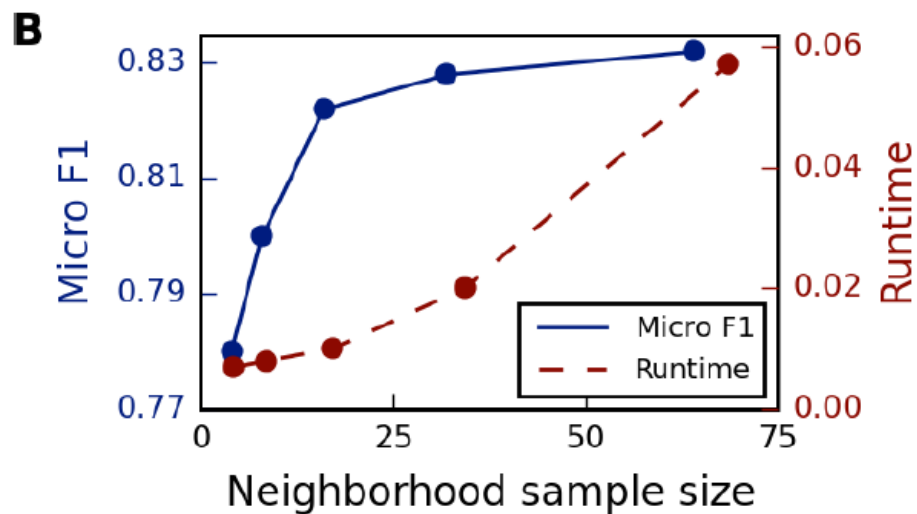
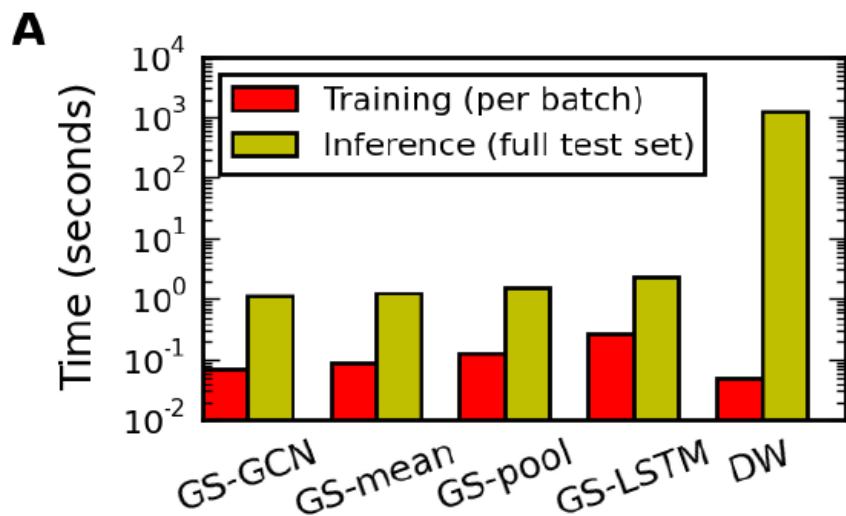
- 实验目的
 - 比较GraphSAGE 相比baseline 算法的提升效果
 - 比较GraphSAGE的不同聚合函数
- 数据集及任务
 - Citation 论文引用网络（节点分类）
 - Reddit 帖子论坛（节点分类）
 - PPI 蛋白质网络（graph分类）
- Baselines
 - Random，随机分类器
 - Raw features，手工特征（非图特征）
 - deepwalk（图拓扑特征）
 - DeepWalk + features， deepwalk+手工特征

- 实验结果1: 分类准确率 (micro-averaged F1 scores)

Table 1: Prediction results for the three datasets (micro-averaged F1 scores). Results for unsupervised and fully supervised GraphSAGE are shown. Analogous trends hold for macro-averaged scores.

Name	Citation		Reddit		PPI	
	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	0.908	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	0.907	0.954	0.482	0.612
GraphSAGE-pool	0.798	0.839	0.892	0.948	0.502	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

- 实验结果2：运行时间和参数敏感性



- 横向对比
 - GraphSAGE是一种能够利用节点的属性信息高效产生未知节点embedding的一种归纳式(inductive)学习的框架
- 纵向对比
 - GraphSAGE采用了采样的机制，克服了其他算法比如GCN训练时内存和显存上的限制，使得图模型可以应用到大规模的图结构数据中，是目前几乎所有工业上图模型的雏形

- 算法的应用领域
 - 在交通、金融、社会科学等有大量相应数据积淀的交叉领域面临着广阔的应用前景
- 未来的发展
 - 扩展GraphSAGE以合并有向图或者多模式图
 - 探索非均匀邻居采样函数

- [1] William L. Hamilton, Zhitao Ying, Jure Leskovec.
Inductive Representation Learning on Large Graphs [C].
NIPS 2017
- [2] GraphSAGE: GCN落地必读论文
<https://zhuanlan.zhihu.com/p/62750137>



知人者智，自知者明。
胜人者有力，自胜者
强。知足者富。强行
者有志。不失其所者
久。死而不亡者，寿。

谢谢！

