

GraphSAGE: GCN 可能没我强

阿泽crz CVer 2020-04-12

点击上方 **CVer** ，选择加 星标 或 置顶
重磅干货，第一时间送达

本文转载自：阿泽的学习笔记

今天看的论文是斯坦福大学的同学的论文《Inductive Representation Learning on Large Graphs》，于 2017 年发表于 NIPS，目前被引次数超过 1200 次。

对于大规模网络图来说，低维的分布编码具有举足轻重的意义，但现有的诸多模型都属于直推式学习（transductive），其可以解决参与训练的节点的编码问题，但无法泛化到未知节点（即，如果有新节点加入需要重新训练）。针对这一痛点，斯坦福大学的同学提出了归纳式学习算法（inductive）——GraphSAGE，该算法可以利用的方式解决了未知节点无法 Embedding 的问题，接下来我们看一下 GraphSAGE 的具体实现过程。

1. Introduction

NetWork Representation 应用广泛，模型可以通过将网络中的节点编码为低维的 Embedding 向量，为下游机器学习任务提供了有效的特征输入，但目前的模型都无法应对网络中的未知节点，属于直推式学习。而在真实场景中，新节点随处可见，如新用户、新帖子、新视频等等。

为了解决这一问题，我们需要一个具有归纳能力模型，它可以利用节点的邻域特征归纳出节点的 Embedding 向量。

相比于直推式学习而言，归纳式学习方式尤其困难。因为要想泛化新节点的 Embedding 向量，则需要模型将新网络与先前的网络对齐，以识别新节点的邻域结构，并捕获节点在图中的局部特征和全局特征。

针对这一痛点，本文作者在 GCN 的基础上提出了 GraphSAGE 算法（SAmple and aggreGatE）用于归纳学习节点的 Embedding 向量，其不仅将 GCN 扩展到无监督的归纳学习任务中，还泛化了 GCN 的聚合函数。

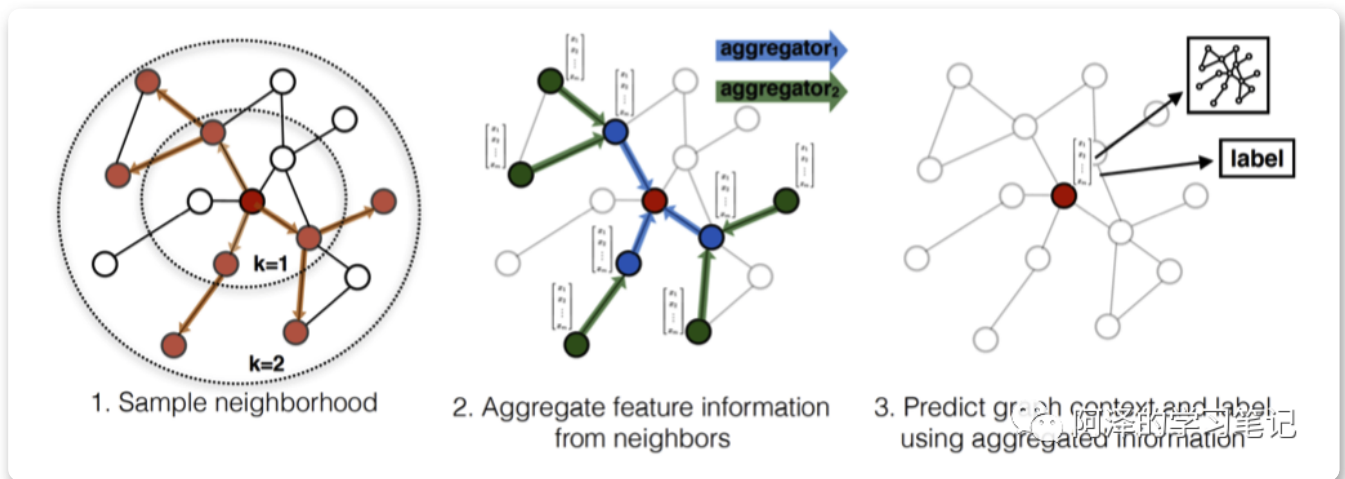
与 GCN 直接学习某个节点的 Embedding 向量不同的是，GraphSAGE **「是利用一组聚合函数进行学习」**。这些聚合函数可以从节点的邻居中学到节点的特征信息，所以即使是新节点也可以通过其邻域信息进行学习。

接下来我们看一下 GraphSAGE 是如何利用聚合函数学习的。

2. GraphSAGE

2.1 Algorithm

我们先概览下 GraphSAGE：



1. 首先对节点的一阶和二阶邻居节点进行采样；
2. 然后根据聚合函数聚合邻居节点的特征；
3. 最后得到节点的 Embedding 向量。

所以 GraphSAGE 大概思想就是聚合邻居信息，然后不断迭代。

放出 GraphSAGE 的伪代码：

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

```



- 第一行是我们要计算的节点的特征输入；
- 第二行是第一个 for 循环遍历深度，可以理解为神经网络的层数；
- 第三行是第二个 for 循环是遍历图中所有节点；
- 第四行是从「**上一层神经网络**」中利用聚合函数聚合当前节点邻居的特征。
- 第五行是将当前节点的特征和邻居特征拼接并经过一个全连接网络得到当前节点的新特征；
- 第七行是归一化；
- 第八行是通过 K 层 GCN 后进行输出。

简单来说就是用 $k-1$ 层的节点的邻居信息和自身信息来更新 k 层的节点信息。这里的聚合函数我们待会再讨论，现在可以默认是一个提取邻居特征的方法。

但这样会出现一个问题：如果我们只是想计算某个新的节点的 Embedding，其实没有必要把整张图的节点的 Embedding 都更新一遍。

针对这个问题，作者给出了算法二：

Algorithm 2: GraphSAGE minibatch forward propagation algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$;
 input features $\{\mathbf{x}_v, \forall v \in \mathcal{B}\}$;
 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 sampling functions, $\mathcal{N}_k : v \rightarrow 2^{\mathcal{V}}, \forall k \in \{1, \dots, K\}$

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

```

1  $\mathcal{B}^K \leftarrow \mathcal{B}$ ;
2 for  $k = K \dots 1$  do
3    $\mathcal{B}^{k-1} \leftarrow \mathcal{B}^k$ ;
4   for  $u \in \mathcal{B}^k$  do
5      $\mathcal{B}^{k-1} \leftarrow \mathcal{B}^{k-1} \cup \mathcal{N}_k(u)$ ;
6   end
7 end
8  $\mathbf{h}_u^0 \leftarrow \mathbf{x}_u, \forall u \in \mathcal{B}^0$ ;
9 for  $k = 1 \dots K$  do
10  for  $u \in \mathcal{B}^k$  do
11     $\mathbf{h}_{\mathcal{N}(u)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_{u'}^{k-1}, \forall u' \in \mathcal{N}_k(u)\})$ ;
12     $\mathbf{h}_u^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_u^{k-1}, \mathbf{h}_{\mathcal{N}(u)}^k))$ ;
13     $\mathbf{h}_u^k \leftarrow \mathbf{h}_u^k / \|\mathbf{h}_u^k\|_2$ ;
14  end
15 end
16  $\mathbf{z}_u \leftarrow \mathbf{h}_u^K, \forall u \in \mathcal{B}$ 

```

阿泽的学习笔记

- 第 1 行到第 7 行是在更新每层神经网络会涉及到的网络节点集合（从最后一层开始更新，应该能知道为什么。）；
- 第 10 行到第 14 行是在遍历需要更新的节点向量，每层要更新的节点集合是在上面已经算出来了；
- 第 13 行有一点小变化，主要是为了方便使用 SGD 进行优化。

这里出现的 $\mathcal{N}_k(u)$ 是指对节点 u 在第 k 层进行邻居采样（每层独立采样）。这里「**邻居采样的大小是固定的**」，以保证每个批处理单元大小都是固定的。

2.2 Loss Function

学习节点的 Embedding 向量是一个非监督学习，我们希望节点与较近的节点的 Embedding 向量相似，而与较远的节点不相似，其损失函数为：

$$J(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u^T \mathbf{z}_v)) - Q \cdot E_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u^T \mathbf{z}_{v_n}))$$

其中，节点 v 是节点 u 在定长随机游走算法中邻居节点； P_n 是负采样分布， Q 定义了负采样的个数。

2.3 Aggregator

对于一个无序网络来说，理想的聚合器是对称的，即：不考虑节点顺序。同时也需要保证 Embedding 向量具有较好的效果。这里作者提出了三种不同的聚合器：

1. [Mean aggregator]

均值聚合器，将邻居节点和当前节点的 Embedding 向量取均值，这与我们先前介绍的 GCN 有很多相似的地方（都是基于邻居节点进行聚合）。

2. [LSTM aggregator]

LSTM 聚合器，因为 LSTM 具有强大的表达能力，但是 LSTM 是非对称的，为了解决这个问题，我们将只需将 LSTM 应用于节点邻居的随机排列，即可使 LSTM 适应无序集合。

3. [Pooling aggregator]

池化聚合器，这是作者最终使用的聚合器。这种聚合方式可以使得节点的每个邻居的 Embedding 向量都可以独立的通过全连接的神经网络，通过这样的转换后最大池化操作可以聚合整个邻居集合。

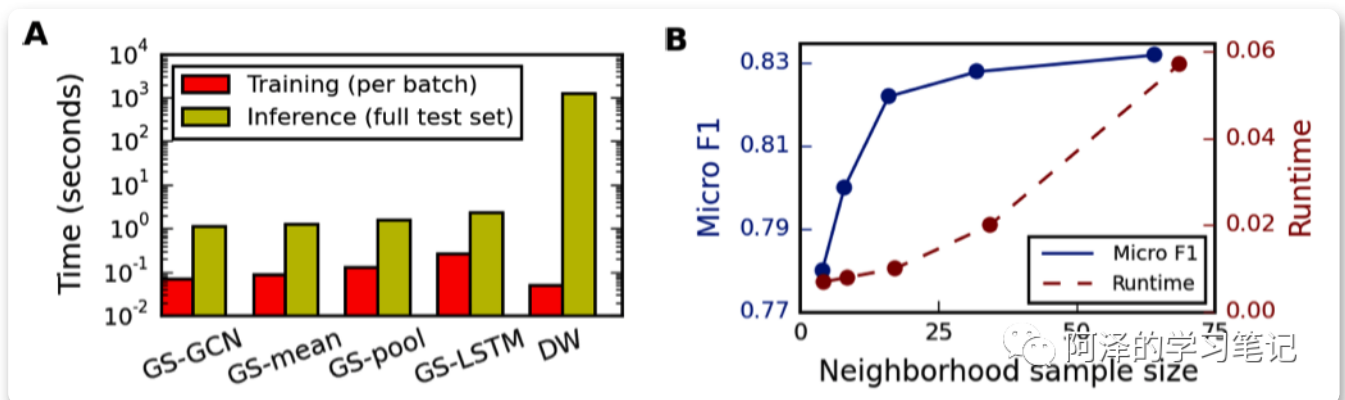
3. Experiments

我们来看一下实验部分。

下表表示 GraphSAGE 的不同聚合器和其他基准算法在不同数据集下的指标，我们看到 LSTM 和 Pooling 聚合器的效果差不多。

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%

下图展示了不同参数聚合器下的模型的运行时间。



4. Conclusion

一句话总结：GraphSAGE 通过聚合节点邻居的特征将 GCN 扩展成归纳学习的方式，解决了未知 Embedding 的难题。此外，GraphSAGE 也支持修改聚合函数，使其更具扩展性。特别是使用 LSTM 和 Pooling 聚合器后，模型的效果得到了显著的提升。

5. Reference

1. 《Inductive Representation Learning on Large Graphs》

重磅！CVer-图神经网络 微信交流群已成立

扫码添加CVer助手，可申请加入CVer-图神经网络 微信交流群，目前已汇集500人！涵盖图神经网络，图卷积网络等。互相交流，一起进步！

同时也可申请加入CVer大群和细分方向技术群，细分方向已涵盖：目标检测、图像分割、目标跟踪、人脸检测&识别、OCR、姿态估计、超分辨率、SLAM、医疗影像、Re-ID、GAN、NAS、深度估计、自动驾驶、强化学习、车道线检测、模型剪枝&压缩、去噪、去雾、去雨、风格迁移、遥感