GraphSAGE阅读笔记

原创 迟博 嬉皮工匠 2020-03-17

收录于话题
#图网络 13 #论文阅读笔记 16

GraphSAGE《Inductive Representation Learning on Large Graphs》阅读笔记

Task：node classification

最近在读GNN的经典文章，网上对这些文章的解读已经非常透彻，本人在阅读文献过程中也学习了各路大神的独到见解，感谢，向你们致敬。在此分享我的阅读心得和热爱。

Abstract

GraphSAGE为解决大多算法在embeddings训练时需要所有nodes参与的问题（对大图不友好）而生。文章提到现有方法是**直推式transductive**的，不能泛化到**未知节点unseen nodes**。GraphSAGE作为一种**归纳式inductive**框架，不仅能够泛化unseen node，还能够通过从node的**局部邻节点**中通过**采样和聚合**学习embeddings，而不是独立训练每个node的embeddings：

“Here we present GraphSAGE, a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data. Instead of training individual embeddings for each node, we learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood”

Introduction

在实际应用场景下需要对未知节点或者整个新图快速的生成嵌入。之前的一些直推式的node embedding方法局限在固定的图（fixed graph），而很多图是不断演变的，这就导致了这些直推方法在新的节点或图出现时要重新训练，效率很低。所以能够适应不断演变的图和节点的归纳能力变得至关重要。作者还提到这种归纳能力应该可以在具有相同特征形式的不同图之间泛化。所谓**transductive**，即节点分类任务中，GCN在训练时是**同时需要训练节点和测试节点**的，但在实际应用中测试节点是不断出现的，所以直推式是不够友好的。而对于大多数机器学习方法或问题，都是**inductive**的，因为训练样本/验证样本/测试样本往往是独立的，显然这种inductive方式更恰当。

归纳式方法难点在于：**边训练边归纳。**

虽然GraphSAGE旨在学习归纳式的节点嵌入方法，但是基于采样的方法使得他们能够掌握图的**结构信息**，用原文的话来讲就是：

“reveal both the node’s local role in the graph, as well as its global position.”

所以，即使是**没有节点特征的图**，GraphSAGE也是适用的。

GraphSAGE通过一组**聚合函数(Aggregator functions)**从节点的**局部邻节点(local neighborhood)**中聚合特征信息。如下图所示：

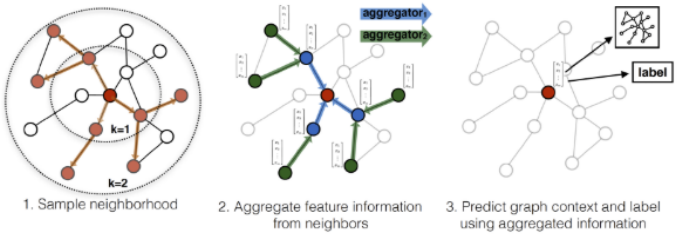

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

大致流程是：每个聚合函数以**不同数量**的**跳数(hops)**或者**搜索深度(search depth)**聚合信息。在test时候，直接应用训练好的聚合函数去处理unseen node。

对于损失函数，GraphSAGE使用了**无监督**方式，而不是任务驱动监督(task-specific supervision)的方式，当然监督的方式也是OK的。

Method

Embedding generation(forward propagation)

首先定义K个聚合函数：

$$AGGREGATE_k, \forall k \in \{1, \dots, K\}$$

以及用于在模型不同层或搜索深度“Search Depth”之间传播信息的权重矩阵：

$$W^k, \forall k \in \{1, \dots, K\}$$

算法流程图：

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

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

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

算法流程所展示的直觉是：方法在每一次迭代过程(search depth)中，节点都会从邻节点中聚合信息。随着迭代次数增多，节点**增量地**获取越来越多的关于图的信息。

K代表着K个聚合函数、K个权重矩阵、以及K层（可以理解为访问到的邻节点的hops）。

1. 采样：在每一次外层循环中，通过采样只选择部分样本进去内层循环。

“instead of iterating over all nodes, we compute **only the representations that are necessary to satisfy the recursion at each depth**”

外层节点样本从**固定数量邻节点*N(v)***中均匀采样得到，而不是所有邻节点。

2. 聚合：每个node将从邻节点(Immediate Neighborhood)中得到的信息聚合为一个vector:

$$\{h_u^{k-1}, \forall u \in \mathrm{N}(v)\} \rightarrow h_{\mathrm{N}(v)}^{k-1}$$

然后将聚合得到的vector和该节点自身的vector **concatenate**到一起，接上一个全连接层，得到的vector作为下一步的输入，记作z_v。至于如何聚合是比较灵活的，作者在后文也有讨论。

Learning the parameters of GraphSAGE

如前文所述，作者设计了无监督损失来优化聚合函数的参数和权重W。

$$J_G(z_u) = -\log(\sigma({z_u}^T z_v)) - Q \cdot E_{v_n \sim P(n)} \log(\sigma({z_u}^T z_n)$$

其中**第一项**使得z_u与其“随机游走Random Walk”邻节点z_v**更相似。**

第二项使得z_u与其非邻居节点“negative samples”**差异化。**

上述的相似性通过向量点积计算得到，*P*，*Q*分别表示negative samples的分布和数量。

另外除了无监督损失，还可以根据特定任务设计task-specific objective.

Three Aggregator Architectures

Mean Aggregator

第一种方式就是elementwise mean，就是将深度搜索到的邻居节点在每一个维度上取平均值。其实这个和GCN很像了，GCN本质上是求和。

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

LSTM Aggregator

这个就是把LSTM用作聚合函数了，但是它本身并不是对称的，也就是并不能保证“**排列不变量**”。所以这里就先随机排列节点的embeddings，然后再扔进LSTM。

Pooling Aggregator

简单来说就是**全连接层+elementwise max-pooling**，具体如下：

$$AGGREGATE_k^{pool} = \max(\{\sigma(W_{pool} h_{u_i}^k + b), \forall u_i \in \mathcal{N}(v)\})$$

Conclusion

本文的核心在于**伪代码的4、5行（聚合）**和关于**sampling**的设计。一方面GraphSAGE的采样方法对实际应用场景中的大图和可变图很友好；另一方面其聚合思想也使得节点特征的学习不再是独立的，而是更多的融合了邻节点的信息。

另外作者也谈到了GraphSAGE与GCN的联系。GCN是直推式的和全局的，每次迭代都需要整张图的邻接矩阵（包括测试节点），GraphSAGE算是对GCN的一种精简。

但是对于固定数量的采样可能有待商榷，我的理解是每个节点对于整张图的贡献度可能是不同的，那么全部去sampling同样的hops可能不够完善，所以GATs大概能解决这个问题吧。当然了，固定数量是为了能够更方便的做concatenation。

个人理解，如有疏漏欢迎批评指正。

参考

https://mp.weixin.qq.com/s/VJn-Sek_aP02MOdcrzbtQA

https://zhuanlan.zhihu.com/p/62750137

推荐阅读

章总的开发心路

服务开关方案-你跑得这么凶，没人拦着怎么行

论文阅读笔记

DIFFPOOL阅读理解

大道至简----多示例学习与注意力机制的巧妙结合

手撸数据结构