GNN 系列（三）：GraphSAGE

原创 金良，杨开漠 Datawhale 2019-08-09

点击上方“**Datawhale**”，选择“星标”公众号

第一时间获取价值内容

【引言】在GCN的博文中我们重点讨论了图神经网络的逐层传播公式是如何推导的，然而，GCN的训练方式需要将邻接矩阵和特征矩阵一起放到内存或者显存里，在大规模图数据上是不可取的。其次，GCN在训练时需要知道整个图的结构信息(包括待预测的节点)，这在现实某些任务中也不能实现(比如用今天训练的图模型预测明天的数据，那么明天的节点是拿不到的)。GraphSAGE的出现就是为了解决这样的问题，这篇文中我们将会详细得讨论它。

Inductive learning v.s. Transductive learning

首先我们介绍一下什么是inductive learning. 与其他类型的数据不同，图数据中的每一个节点可以通过边的关系利用其他节点的信息，这样就产生了一个问题，如果训练集上的节点通过边关联到了预测集或者验证集的节点，那么在训练的时候能否用它们的信息呢？如果训练时用到了测试集或验证集样本的信息(或者说，测试集和验证集在训练的时候是可见的)，我们把这种学习方式叫做transductive learning，反之，称为inductive learning. 显然，我们所处理的大多数机器学习问题都是inductive learning，因为我们刻意的将样本集分为训练/验证/测试，并且训练的时候只用训练样本。然而，在GCN中，训练节点收集邻居信息的时候，用到了测试或者验证样本，所以它是transductive的。

概述

GraphSAGE是一个inductive框架，在具体实现中，训练时它仅仅保留训练样本到训练样本的边。inductive learning 的优点是可以利用已知节点的信息为未知节点生成Embedding. GraphSAGE 取自 Graph SAmple and aggreGatE, SAmple指如何对邻居个数进行采样。aggreGatE指拿到邻居的embedding之后如何汇聚这些embedding以更新自己的embedding信息。下图展示了GraphSAGE学习的一个过程：

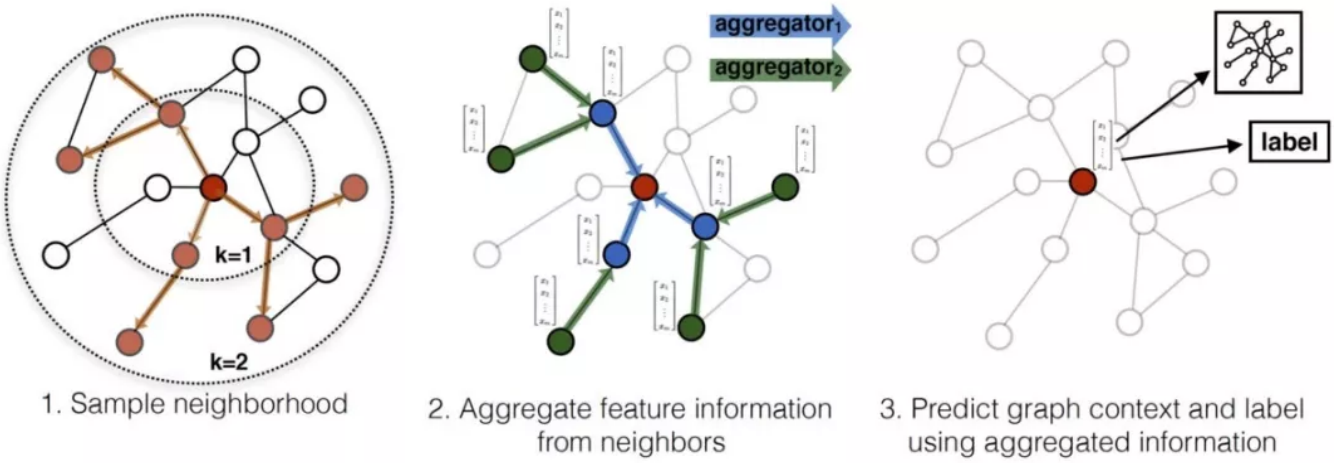

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

1.对邻居采样

2.采样后的邻居embedding传到节点上来，并使用一个聚合函数聚合这些邻居信息以更新节点的embedding

3.根据更新后的embedding预测节点的标签

算法细节

1. 节点 Embedding 生成(即：前向传播)算法

这一节讨论的是如何给图中的节点生成(或者说更新)embedding, 假设我们已经完成了GraphSAGE的训练，因此模型所有的参数(parameters)都已知了。具体来说，这些参数包括K个聚合器$\mathbf{AGGREGATE}_k, \forall k \in \{1, \ldots, K\}$(见下图算法第4行)中的参数, 这些聚合器被用来将邻居embedding信息聚合到节点上，以及一系列的权重矩阵$\mathbf{W}^k, \forall k \in \{1, \ldots, K\}$(下图算法第5行), 这些权值矩阵被用作在模型层与层之间传播embedding的时候做非线性变换。

下面的算法描述了我们是怎么做前向传播的：

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 $v \in \mathcal{V}$

1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
2 **for** $k = 1...K$ **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 $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

算法的主要部分为：（line 1)初始化每个节点embedding为节点的特征向量 (line 3)对于每一个节点v (line 4)拿到它采样后的邻居的embedding $h_u, u \in \mathcal{N}(v)$并将其聚合，这里$\mathcal{N}(v)$表示对邻居采样 (line 5)根据聚合后的邻居embedding($h_{\mathbf{N}(v)}$)和自身embedding(h_v)通过一个非线性变换($\sigma(\mathbf{W} \cdot \square)$更新自身embedding.

算法里的K这个比较难理解，下面单独来说他，K之前提到过，它既是聚合器的数量，也是权重矩阵的数量，还是网络的层数，这是因为每一层网络中聚合器和权重矩阵是共享的。网络的层数可以理解为需要最大访问到的邻居的跳数(hops)，比如在figure 1中，红色节点的更新拿到了它一、二跳邻居的信息，那么网络层数就是2。为了更新红色节点，首先在第一层($k = 1$)我们会将蓝色节点的信息聚合到红色节点上，将绿色节点的信息聚合到蓝色节点上。在第二层($k = 2$)红色节点的embedding被再次更新，不过这次用的是更新后的蓝色节点embedding，这样就保证了红色节点更新后的embedding包括蓝色和绿色节点的信息。

2. 采样 (Sample) 算法

GraphSAGE采用了定长抽样的方法，具体来说，定义需要的邻居个数S, 然后采用有放回的重采样/负采样方法达到S,。保证每个节点(采样后的)邻居个数一致是为了把多个节点以及他们的邻居拼成Tensor送到GPU中进行批训练。

3. 聚合器 (Aggregator) 架构

GraphSAGE 提供了多种聚合器，实验中效果最好的平均聚合器(mean aggregator)，平均聚合器的思虑很简单，每个维度取对邻居embedding相应维度的均值，这个和GCN的做法基本一致(GCN实际上用的是求和)：

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

举个简单例子，比如一个节点的3个邻居的embedding分别为 $[1,2,3,4]$, $[2,3,4,5]$, $[3,4,5,6]$ 按照每一维分别求均值就得到了聚合后的邻居embedding为 $[2,3,4,5]$.

论文中还阐述了另外两种aggregator: **LSTM aggregator** 和 **Pooling aggregator**, 有兴趣的可以去论文中看下。

4. 参数学习

到此为止，整个模型的架构就讲完了，那么GraphSAGE是如何学习聚合器的参数以及权重变量$\mathbf{W}$的呢？在有监督的情况下，可以使用每个节点的预测label和真实label的交叉熵作为损失函数。在无监督的情况下，可以假设相邻的节点的输出embeding应当尽可能相近，因此可以设计出如下的损失函数：

$$J_{\mathcal{G}}\left(\mathbf{z}_u\right)=-\log\left(\sigma\left(\mathbf{z}_u^{\top}\mathbf{z}_v\right)\right)-Q\cdot\mathbb{E}_{v_n\sim P_n(v)}\log\left(\sigma\left(-\mathbf{z}_u^{\top}\mathbf{z}_{v_n}\right)\right)\quad(2)$$

其中z_u是节点u的输出embedding，v是节点u的邻居（这里邻居是广义的，比如说如果v和u在一个定长的随机游走中可达，那么我们也认为他们相邻），P_n是负采样分布，Q是负采样的样本数量，所谓负采样指我们还需要一批不是v邻居的节点作为负样本，那么上面这个式子的意思是相邻节点的embedding的相似度尽量大的情况下保证不相邻节点的embedding的期望相似度尽可能小。

后话

GraphSAGE采用了采样的机制，克服了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。然而，每个节点这么多邻居，采样能否考虑到邻居的相对重要性呢，或者我们在聚合计算中能否考虑到邻居的相

对重要性？这个问题在我们的下一篇博文Graph Attentioin Networks中做了详细的讨论。

阅读原文

喜欢此内容的人还喜欢

如何阅读源码

是不是很酷