GraphSAGE理论与实践

# 前言

GraphSAGE是2017年提出的一种图神经网络算法，论文题目是《Inductive Representation Learning on Large Graphs》。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。GraphSAGE解决了GCN网络的局限性: GCN训练时需要用到整个图的邻接矩阵，依赖于具体的图结构，一般只能用在直推式学习（Transductive Learning）。GraphSAGE使用多层聚合函数，每一层聚合函数会将节点及其邻居的信息聚合在一起得到下一层的特征向量，GraphSAGE采用了节点的邻域信息，不依赖于全局的图结构。在工业界一直受到重视。

图神经网络的任务一般有Transductive (直推式)和Inductive(归纳式)。Transductive通常指要预测的节点在训练时已经出现过，例如有一个作者关系网络，知道部分作者的类别，用整个网络训练GCN，最后预测未知类别的作者。Inductive指要预测的节点在训练时没有出现，例如用今天的图结构训练，预测明天的图。

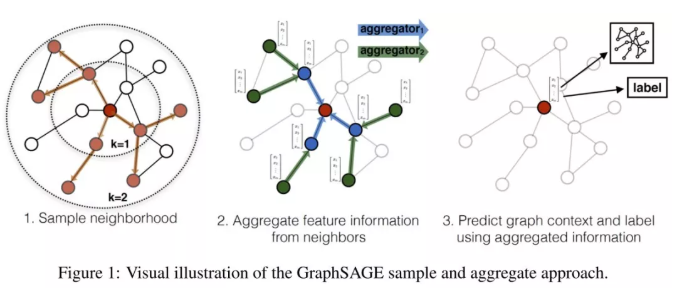
在GraphSAGE之前，图神经网络主要的方法有DeepWalk，GCN等，这些GNN方法的不足在于需要对全图进行学习，而且是以Transductive learning为主，也就是说需要在训练的时候，图就已经包含了要预测的节点。训练时不包含的节点，预测时也无法给出预测结果。

在实际应用中，图的结构会频繁变化，在预测阶段，可能会往图中新添加新节点。针对这种情形，采用直推式的模型（比如DeepWalk，GCN等就无法满足了）。GraphSAGE就是针对这种场景提出的，GraphSAGE = Graph Sample Aggregate，即对图进行采样（sample）和聚合（aggregate）。

# GraphSAGE

## GraphSAGE基本思路

GraphSAGE 包含采样和聚合 (Sample and aggregate)，首先使用节点之间连接信息，对邻居进行采样，然后通过多层聚合函数不断地将相邻节点的信息融合在一起。用融合后的信息预测节点标签。下图展示了 GraphSAGE 的聚合过程，采用了两层聚合层。



以上图为例进行说明GraphSAGE采样和聚合的过程：

在第一幅图是采样（sample）的过程。在这个图中，需要对最中心的节点进行embedding更新，先从与它相邻的节点中选择S1个（图中选择了3个）节点。这里假设k=2，那么我们还需要对第二层再进行采样，也就是对刚才选择的S1个邻居节点，再选择他们的邻居节点。

在第二幅图中进行aggregate的过程，也就是先拿邻居节点的邻居更新邻居的信息，再用更新后的邻居的信息来更新目标节点（也就是最中心的节点）的信息。

在第三幅图中进行节点预测，假如要预测一个未知节点的信息，只需要用它的邻居节点的信息来预测即可。

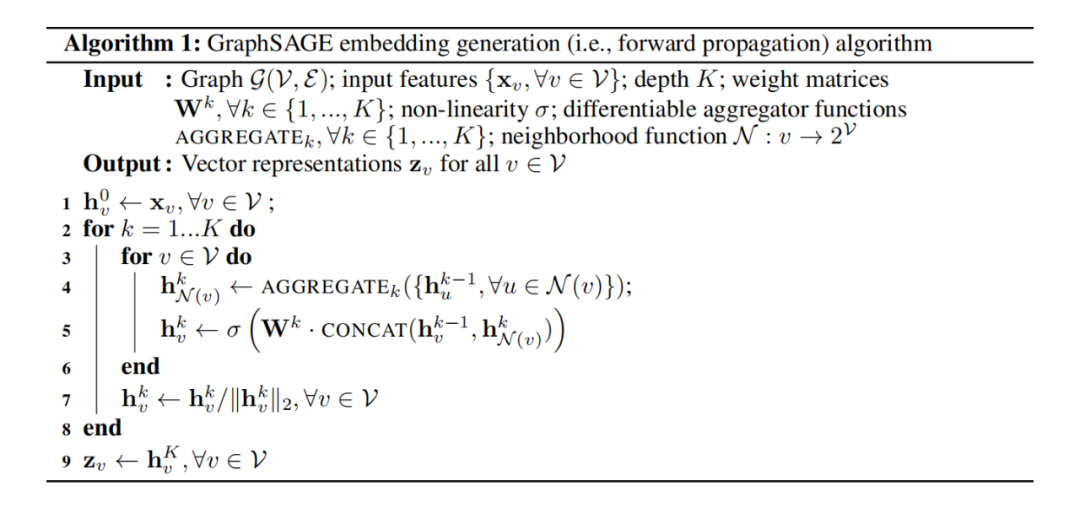
举例：如果我想知道小明是一个什么性格的人，可以找几个他关系好的小伙伴观察一下，为了进一步确认，再选择他的小伙伴们的其他小伙伴，再观察一下。也就是通过小明的小伙伴们的小伙伴，来判断小明的小伙伴们是哪一类人，然后再根据他的小伙伴们，就可以粗略的得知，小明是哪一类性格的人了。

## GraphSAGE算法流程

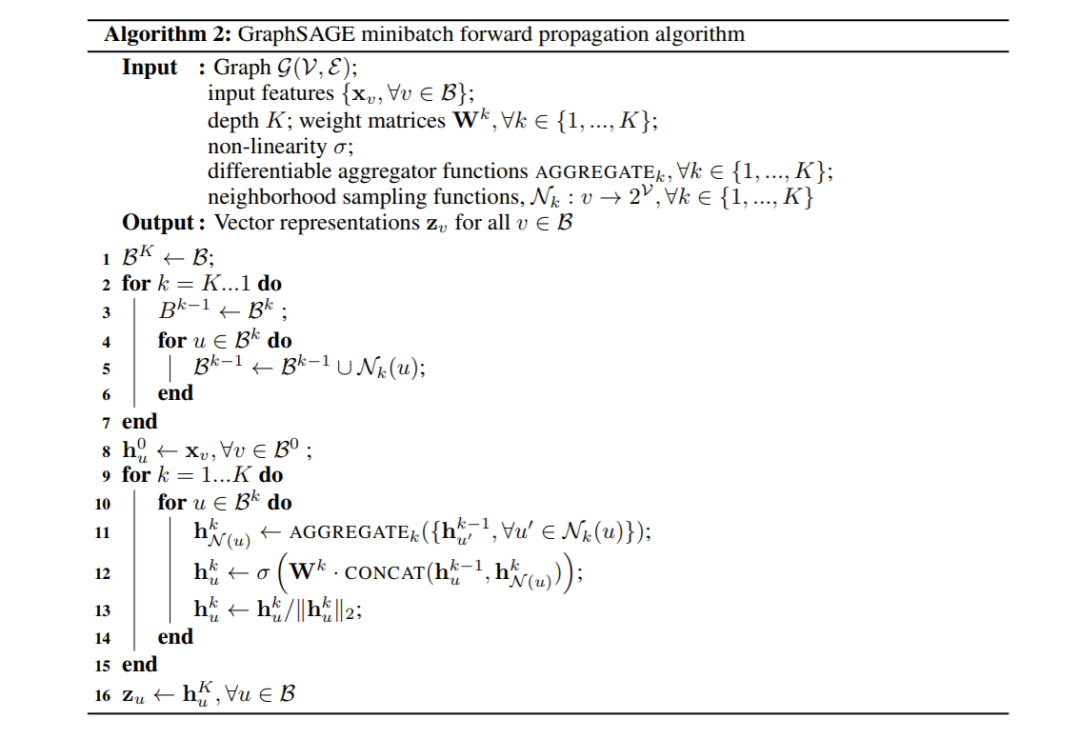
总体来说，GraphSAGE生成（更新）节点embedding的过程其实是假设已经完成了GraphSAGE的训练，因此模型的所有参数都已知了。

具体来说，这些参数包含K个聚合器AGGREGATEk（见下图第4行）中的参数，这些聚合器被用于将邻居节点的embedding信息聚合到节点上，以及一系列的权重矩阵W（见下图第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。

下面是GraphSAGE产生embedding的过程，也被称为前向传播过程：



伪代码中的h0表示节点v的初始特征向量，包含K层聚合操作。在第k次聚合生成v节点特征向量时，会采用聚合函数把v节点的邻居信息融合在一起。这一操作也可改成 minibatch的，伪代码如下：



上面的伪代码中，B = BK为要生成向量的节点集合，Bk-1是深度为1的邻域，B0为深度为K的邻域，B0包含的节点最多。Nk(v)表示v节点在第k次聚合时的邻域，节点在每一层的邻域数量都不同，通过采样得到。

算法的第7行，其实就是一个sample的过程，并且将sample的结果保存到B中。接下来的9-15行，就是一个aggregate的过程，按照前面sample的结果，将对应的邻居信息 aggregate 到目标节点上来。

仔细观察，可以发现sample的过程是从K到1的（看第2行），而aggregate的过程是从1到K的（第9行）。这是因为采样的时候，先从整张图选择目标节点，然后对这些节点的邻居进行采样，并且逐渐采样到远一点的邻居上。而在聚合时，是先从最远处的邻居上开始聚合，最后第K层的时候，才能聚合到目标节点上来。

在图神经网络中，数据集都通常非常大，首先mini batch的思路非常重要，然后在GraphSAG中，只需要对自己采样的数据进行聚合，无需考虑其它节点。每个batch可以是一批sample结果的组合。这是GraphSAGE的精妙之处之一。

## GraphSAGE聚合函数

在GraphSAGE中，聚合函数非常重要。关于聚合函数的选择有两个条件：

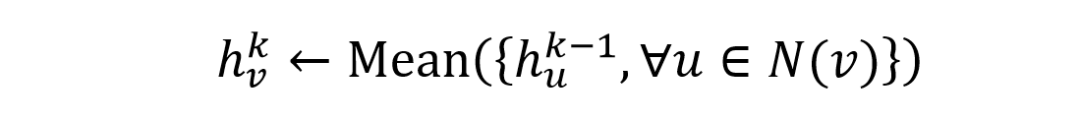
* 可导，因为要反向传递来训练目标的聚合函数参数；
* 对称，这里的对称指的是对输入不敏感，因为在聚合的时候，图中的节点关系并没有顺序上的特征。

所以在作者原文中选择的都是诸如 Mean，max pooling 之类的聚合器，虽然作者也使用了LSTM，但是在输入前会将节点进行shuffle操作，也就是说LSTM从序列顺序中并不能学到什么知识。

GraphSAGE提供了四种聚合节点的函数：

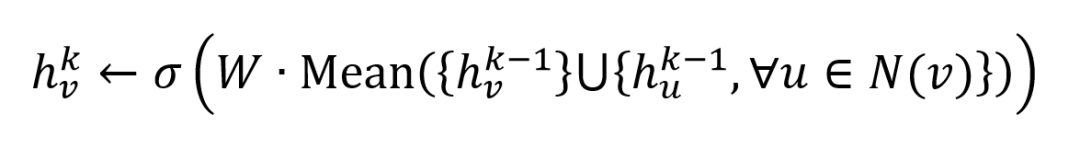
### Mean aggregator

Mean aggregator: 对节点 v 进行聚合时，对节点 v 和邻域的特征向量求均值。



### GCN aggregator

GCN aggregator: 采用了类似GCN卷积的方式进行聚合，公式和Mean aggregator类似：



### LSTM aggregator

LSTM aggregator: 作者任务LSTM有比较好的抽取特征能力，因此也使用了LSTM进行聚合，但是因为节点之间没有明显的顺序关系，因此会打乱之后放入LSTM。

### Pooling aggregator

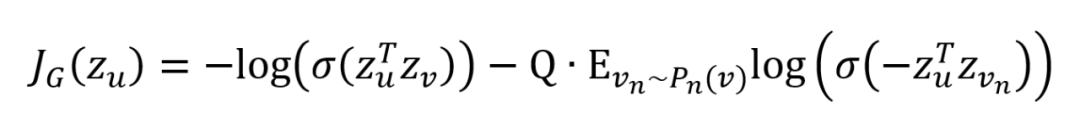
Pooling aggregator: 先把所有邻居节点的特征向量传入一个全连接层，然后使用 max-pooling聚合。



## GraphSAGE训练

GraphSAGE可以采用无监督训练或者有监督训练。其实不论选择哪种训练方式，我们的目的还是使用它来完成节点embedding的过程。

无监督训练采用负采样算法。也就是利用图中的邻居关系，当两个节点距离相近时默认这两个节点的embedding也相似；相反，如果两个节点的相距较远，那么它们的embedding应该差异很大。基于此，采用无监督训练时损失函数公式如下：



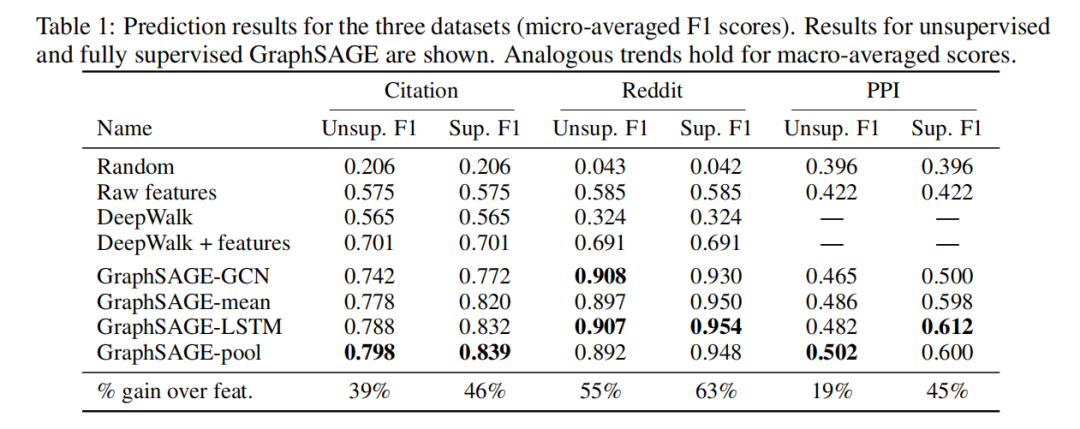
公式中的zu是经过GraphSAGE聚合之后的特征向量，节点v是节点u邻域内的节点，v\_n表示不是节点u的邻居，Q表示负采样数量。

在论文中，作者对邻居的定义是：直接使用DeepWalk进行随机游走，步长为 5，测试50次，走得到的都是邻居。

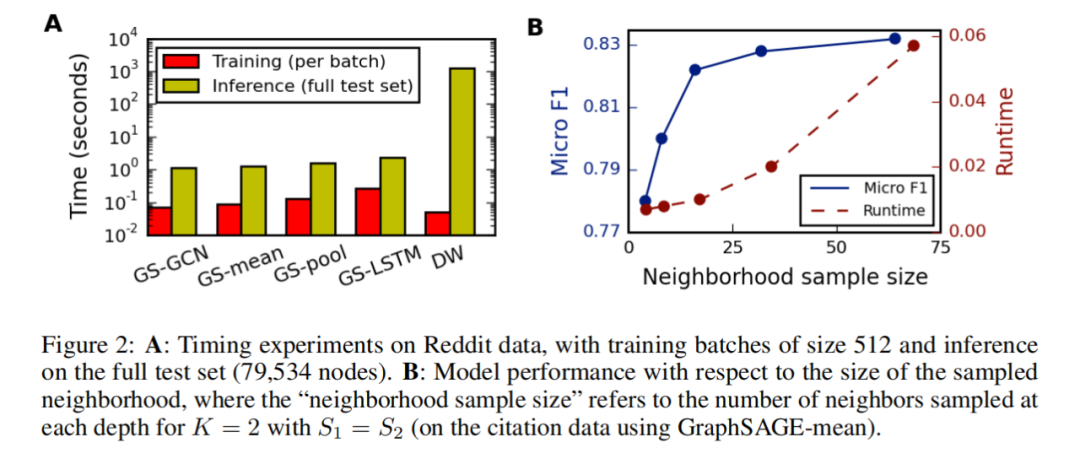
对于有监督训练可以使用任务相关的目标函数，例如节点分类时采用交叉熵损失函数。

# 实验

作者对比了不同算法的性能，也对比了GraphSAGE四种聚合方式的效果，如下表所示。



下图A是训练和测试时间的实验结果，B是采样邻域大小对性能影响的结果。



# 总结

GraphSAGE的精华在于inductive learning，inductive learning可以在测试时对新加入的节点进行推理，这在实际场景的应用中是非常重要的。

参考：

<https://mp.weixin.qq.com/s/pn4GYn9CU-sq49qg2OHkIw>

<https://mp.weixin.qq.com/s/4XhOp8AQUfebLCeAK0xVcQ>

<https://mp.weixin.qq.com/s/1DHvLLysMU24dBeLzbSpUA>