GAT:Graph Attention Networks

1.1 图的定义

对于任意一个顶点 i，它在图上邻居 $\mathcal{N}_i$,构成第一种特征，即图的结构关系。

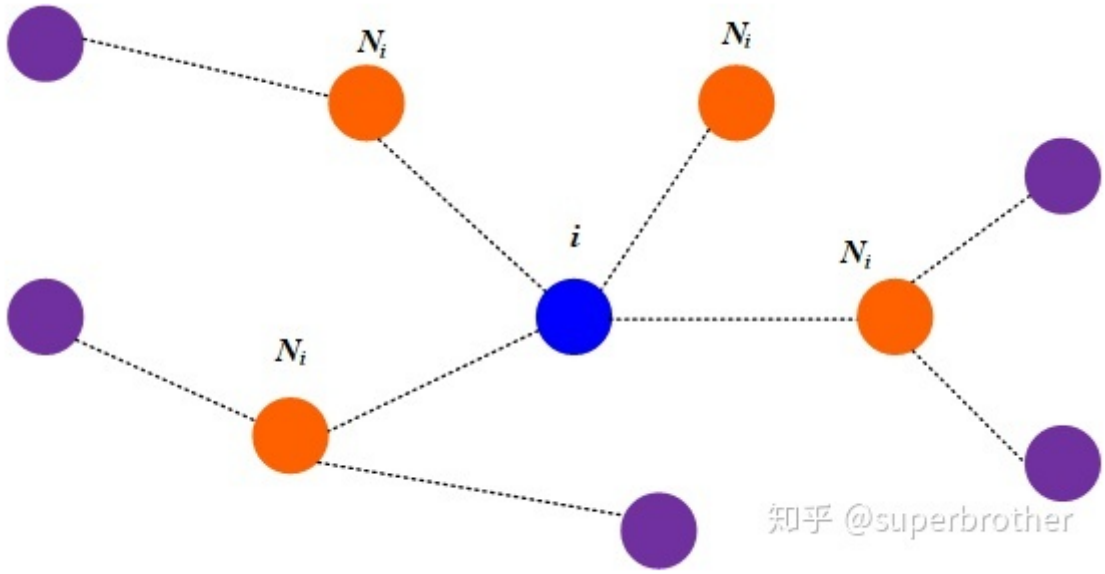

图1 graph示意图

每个顶点还有自己的特征 h_i （通常是一个高维向量）。

$h_i = [h_{i1}, h_{i2}, h_{i3} \dots h_{in}]$

1.2 GAT的计算方式

[补充]GCN的局限性

GCN是处理transductive任务的一把利器（transductive任务是指：训练阶段与测试阶段都基于同样的图结构），然而GCN有**两大局限性**是经常被诟病的：

（a）**无法完成inductive任务，即处理动态图问题。**inductive任务是指：训练阶段与测试阶段需要处理的graph不同。通常是训练阶段只是在子图（subgraph）上进行，测试阶段需要处理未知的顶点。（unseen node）

（b）**处理有向图的瓶颈，不容易实现分配不同的学习权重给不同的neighbor。**这一点在前面的文章中已经讲过了，不再赘述，如有需要可以参考下面的链接。

- **Global graph attention**

就是每一个顶点 i 都对于图上任意顶点都进行attention运算。可以理解为图1的蓝色顶点对于其余全部顶点进行一遍运算。

优点：完全不依赖图的结构，对于inductive任务无压力

缺点：相当于丢弃了图结构的特征，效果可能会很差，并且面临高昂的运算成本

- **Mask graph attention**

注意力机制的运算只在邻居顶点上进行，也就是说图1的蓝色顶点只计算和橙色顶点的注意力系数。

原文：*"We inject the graph structure into the mechanism by performing masked attention—we only compute eij for nodes j ∈Ni, whereNi is some neighborhood of node i in the graph. "*

2. GAT详解

分为两个step

2.1 计算注意力系数（attention coefficient）

对于顶点i，逐个计算它的邻居们($j \in N_i$)和顶点i之间的相似系数

$e_{ij} = a([W \cdot h_i || W \cdot h_j]), j \in N_i \quad (1)$

公式解读：W为共享参数，$W \cdot h_i$相当于对顶点的特征进行增维，这是一种常见的特征增强方法(feature augment)，$||$表示拼接(concatenate), 最后用$a(\cdot)$把拼接后的高维特征映射到一个实数上，作者是通过single layer feedforward neural network实现的。

==》显然学习顶点i, j之间的相关性就是通过参数W和映射函数$a(\cdot)$实现的。

接着再用softmax将相关性系数归一化 就得到了注意力权重

$$\alpha_{ij} = \frac{exp\left(LeakyReLU(e_{ij})\right)}{\sum_{k \in \mathcal{N}_i} exp\left(LeakyReLU(e_{ik})\right)} \qquad (2)$$

至于为何用LeakyRelu未知

2.2 加权求和（aggregate）

根据计算好的注意力系数，把特征加权求和（aggregate）一下

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W h_j\right) \qquad (3)$$

h_i' 是h_i更新了以后的新特征，融合了每个邻结点的信息，$\sigma(\cdot)$是sigmoid激活函数、

还可以使用Multi-head的attention增强表达。

$$h_i'(K) = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k h_j\right) \qquad (4)$$

[k表示head的个数，不同的head]

3.与GCN的联系与区别

本质上而言：**GCN与GAT都是将邻居顶点的特征聚合到中心顶点上（一种aggregate运算），**利用graph上的local stationary学习新的顶点特征表达。**不同的是GCN利用了拉普拉斯矩阵，GAT利用attention系数。**一定程度上而言，GAT会更强，因为 顶点特征之间的相关性被更好地融入到模型中。

3.2 为什么GAT适用于有向图?

我认为最根本的原因是GAT的运算方式是逐顶点的运算（node-wise），这一点可从公式（1）—公式（3）中很明显地看出。每一次运算都需要循环遍历图上的所有顶点来完成。逐顶点运算意味着，摆脱了拉普利矩阵的束缚，使得有向图问题迎刃而解。

3.3为什么GAT适用于inductive任务?

GAT中重要的学习参数是 W 与 $a(\cdot)$，因为上述的逐顶点运算方式，这两个参数仅与1.1节阐述的顶点特征相关，与图的结构毫无关系。所以测试任务中改变图的结构，对于GAT影响并不大，只需要改变 $\mathcal{N}_i$ ，重新计算即可。

与此相反的是，**GCN是一种全图的计算方式，一次计算就更新全图的节点特征。学习的参数很大程度与图结构相关，这使得GCN在inductive任务上遇到困境。**