图注意力网络

Graph Attention Networks

**提出**

栅格化的数据结构可以很便捷的使用有可学习参数的局部滤波器，然后对每个位置滤波，但非栅格化的数据不行，如3D点云、社交网络等。这些数据可以以图的形式展示。先前的GNN通常需要不断迭代直到收敛。然而，现在提出了把“卷积”这个概念泛化到图领域的方法，分为基于谱的方法和不基于谱的方法。

谱方法通常需要傅里叶变化、特征分解、拉普拉斯矩阵等，难以计算。谱方法是基于特定的图结构的，不能被用于另一个结构的图。另一方面，非谱的方法把“卷积”作用在空间上相邻的节点。这种方法的挑战在于如何将卷积作用在不同个数的邻居上，同时还要保持共享权重这一性质。可以对不同度的节点设计不同的参数，或者GraphSAGE固定采样邻居数，然后做聚合操作。Attention已经快成为和序列有关的任务的*de facto*了。当attention机制用在单个序列上，我们叫做intra/self-attention。

基于这些我们提出了基于注意力的节点分类方法：计算每个节点的隐藏表示，再用自注意力机制来注意邻居节点。注意力框架：可以在节点的邻居上并行计算；可以被用于不同度的节点上；可以泛化到没见过的图。

**GAT结构**

输入：节点的特征h，N代表总结点数，F代表特征维数

输出：新的节点特征，特征维数可能不同

为了有足够将输入映射到更高层特征的表达能力，至少需要一个可以学习的线性变化，即权重矩阵。然后我们在节点上使用（共享的）自注意力机制，注意力系数：

指示了节点j的特征对节点i有多重要。在最一般的情形下，模型允许节点注意到图中所有的节点。所以我们引入注意力遮挡（mask attention），只注意节点的一阶邻居（包括节点自己）。最后使用softmax归一化系数。我们提出的注意力机制a就是简单的一层前馈神经网络，参数为，然后加一个Leaky ReLU添加非线性性。

最后，线性组合得到新特征

为了让训练过程更平稳，我们引入多头注意力机制，即K个独立的上式，最后再把特征拼接起来。

不拼接的话也可以用平均、求和等聚合操作。

程序中：

1. kernel   # W in the paper (F x F')
2. attention\_kernel   # Attention kernel a in the paper (2F' x 1)
3. # Compute inputs to attention network
4. features = tf.matmul(input\_feature, kernel)  # (N x F')
6. # Compute feature combinations
7. # Note: [[a\_1], [a\_2]]^T [[Wh\_i], [Wh\_2]] = [a\_1]^T [Wh\_i] + [a\_2]^T [Wh\_j]
8. attn\_for\_self = tf.matmul(features, attention\_kernel[0])    # (N x 1)
9. attn\_for\_neighs = tf.matmul(features, attention\_kernel[1])  # (N x 1)
11. # Attention head a(Wh\_i, Wh\_j) = a^T [[Wh\_i], [Wh\_j]]
12. dense = attn\_for\_self + tf.transpose(attn\_for\_neighs)  # (N x N)
14. # Add nonlinearty
15. dense = tf.nn.leaky\_relu(dense)
16. # Mask values before activation (Vaswani et al., 2017)
17. mask = 10e9 \* tf.sparse.add(-tf.ones([num\_of\_nodes, num\_of\_nodes], dtype=tf.float32), adjacency)
18. dense += mask
20. # Apply softmax to get attention coefficients
21. dense = tf.nn.softmax(dense)  # (N x N)
23. # Linear combination with neighbors' features
24. node\_features = tf.matmul(dropout\_attn, dropout\_feat)  # (N x F')

优点：

计算高效，自注意力层可以在每个边上并行计算，输出也可以并行计算。

给邻居施加了不同的重要性，同时有利于解释性。

注意力参数全图共享，不需要全局结构信息或者所有节点的特征信息。

可以自由推广到有向图，即只有当边j→i存在时计算

可以用于推断训练中未见的结构

和邻居节点的顺序无关，因为并行了。不需要重复多次随机序列再LSTM。

缺点：

稀疏矩阵暂时不能用batch加速，稀疏矩阵也不太好用GPU加速。

感受野的上界就是图的深度，可以用残差网络来加深网络。

边上并行计算可能有重复，因为邻居的特征通常比较相似。

实验

数据集：

直推式学习（所有特征在训练中都出现过）。选择了Cora，Citeseer和Pubmed数据集，都是论文引文数据集，节点是论文，边（无向）是引用。节点的特征是论文的词袋表示。每个节点有一个类别，每类只用20个节点训练，但是训练时用到了所有的节点（注意力机制用到了所有节点，优化时类似GCN的半监督学习）。

推断式学习（测试集在训练时没出现过）。Protein-Protein interaction(PPI)数据集，包含了人体不同的组织。训练集有20张图，验证和测试集各2张。节点特征为某个位置的基因特征，每个节点都拥有若干个标签（共121类）。

SOTA模型：直推式GCN，推断式GraphSAGE及其变种。再单独用一个MLP，直接对节点特征来预测。

超参数：

直推：K=8，八个多头注意力，F’=8特征维度。激活函数为ELU。最后一层为特征维度为C（类别数）的单头注意力，加一个softmax激活函数用于预测类别。

推断：K=4，F’=256，最后一层为K=6(类别数)头注意力机制，还加上了跳跃连接。