基于残差机制的图卷积神经网络

1. 算法介绍

基于现有的GCN网络架构，设置两层卷积层为隐藏层，接受图结构G=<V,E>及图的顶点特征矩阵 作为输入，针对赛事要求，即对给定数据集，在不损失计算精度（计算的中间过程及其最后结果应全部采用32位浮点数精度）的情况下，以尽可能短的时间完成GCN推理的计算。我们引入残差机制，在每一个卷积层的输出中加入上一阶段的节点特征，从而起到自特征凸显作用。同时，为控制自特征的影响程度，我们引入一个超参数调节特征值。

1. 设计思路

按照赛事要求，要保证计算精度，且要尽可能短的时间完成，那么就不能加入新的计算量。而计算时间主要集中在对节点信息的聚合过程中，该过程由固定公式确定，因此我们转换思路，提高模型的计算精度，且不改变原有的计算时间。而经过调研后发现，节点自身的特征才是最为重要的特征，因此，我们想到了残差机制，并将其应用在节点特征中。但同时想到自身特征的影响不应该大于节点聚合的特征，因此，我们引入一个超参数作为控制。

1. 算法优化

对于GCN的算法优化体现在节点特征的增强方面，引入残差机制，并设置控制系数。实现相同时间内的高精度推导。

1. 详细算法设计与实现

全局变量和数据结构

* v\_num 和 e\_num: 顶点和边的数量。
* F0, F1, F2: 各层特征维度。
* edge\_index 和 edge\_val: 邻接列表和边权值。
* degree: 每个顶点的度数。
* raw\_graph: 原始图数据。

数据读取函数

* readGraph(char \*fname): 从指定文件中读取顶点和边的信息，构建原始图数据。
* readFloat(char \*fname, float \*&dst, int num): 从指定文件中读取浮点数数据，存储到 dst 中。

数据预处理

* raw\_graph\_to\_AdjacencyList(): 将原始图数据转换为邻接列表形式，并计算每个顶点的度数。
* edgeNormalization(): 根据度数对边权值进行归一化处理。

初始化和矩阵操作

* initFloat(float \*&dst, int num): 初始化浮点数数组 dst，并将其值设为0。
* XW(int in\_dim, int out\_dim, float \*in\_X, float \*out\_X, float \*W): 实现特征矩阵 in\_X 与权重矩阵 W 的矩阵乘法，结果存储在 out\_X 中。
* AX(int dim, float \*in\_X, float \*out\_X): 实现邻接矩阵 A 与特征矩阵 in\_X 的矩阵乘法，结果存储在 out\_X 中。
* ReLU(int dim, float \*X): 对输入矩阵 X 应用ReLU激活函数。
* LogSoftmax(int dim, float \*X): 对输入矩阵 X 应用LogSoftmax激活函数。
* MaxRowSum(float \*X, int dim): 计算输入矩阵 X 每行的和，并返回最大值。

内存释放

* freeFloats(): 释放所有动态分配的浮点数数组内存。

残差块函数

* residualBlock(int in\_dim, int out\_dim, float \*in\_X, float \*out\_X, float \*W, bool apply\_relu): 实现带有残差连接的网络层，将输入 in\_X 经过权重矩阵 W 处理后，加入残差连接，结果存储在 out\_X 中。

主函数

* main(int argc, char \*\*argv): 读取输入参数，初始化数据，调用各模块进行计算，输出结果并计时，最后释放内存。