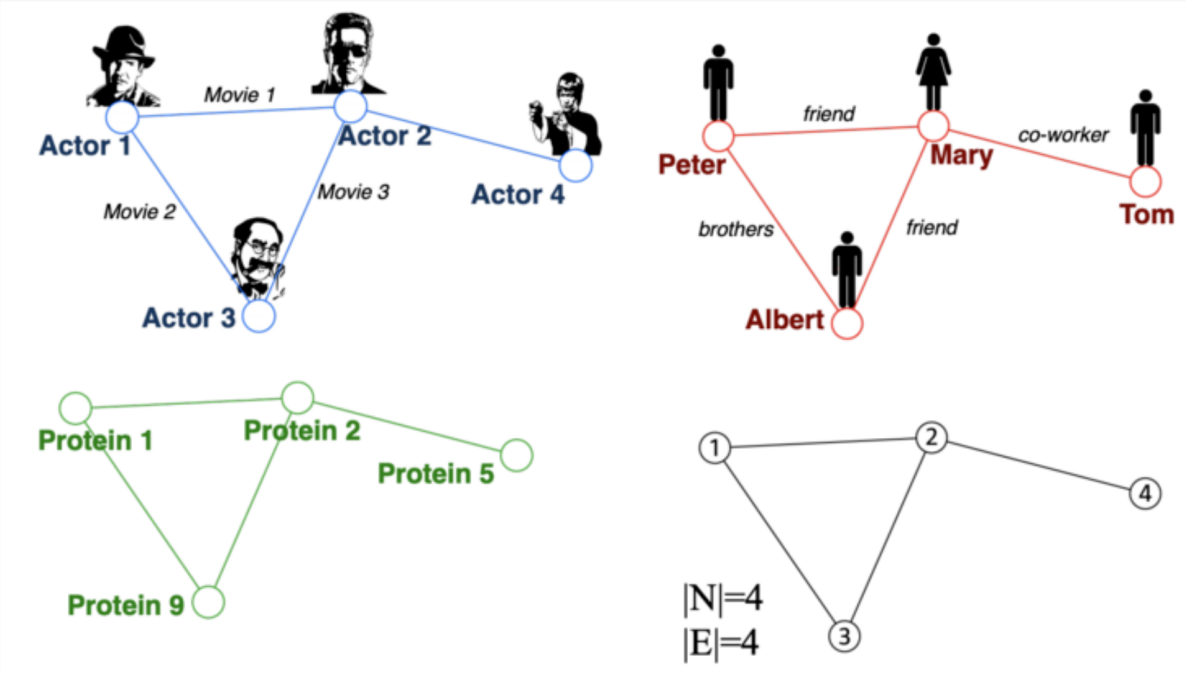
题目 图神经网络计算的优化

1 背景

1.1 图

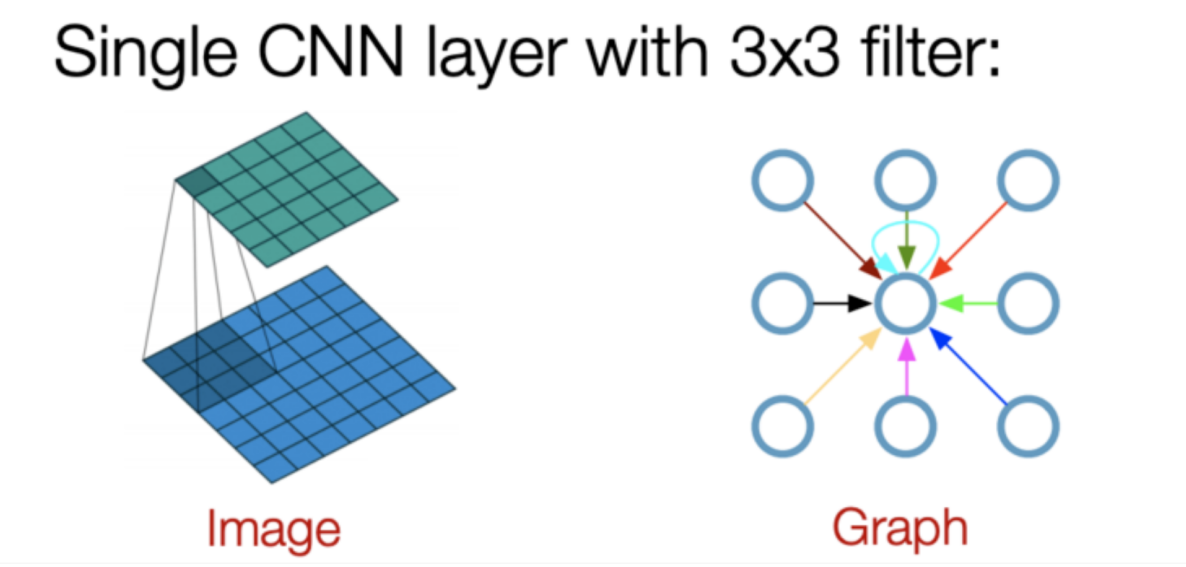
图（Graph）是一种普遍存在的数据结构，在我们的世界里，看到的很多数据本质上都是图，比如分子、社交网络、论文引用网络等。由于图的不规则性，在大规模的场景中很难被快速处理，因此关于图的计算引起了新的热点。



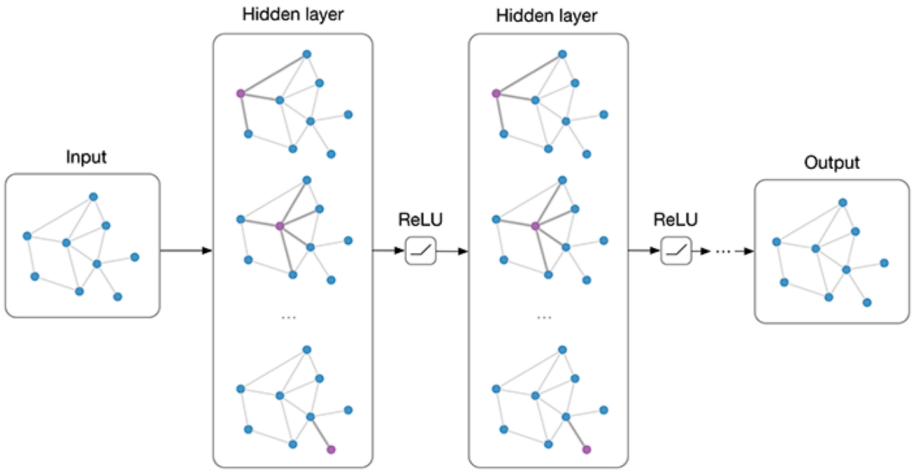
1.2 图卷积神经网络

图卷积神经网络（Graph Convolutional Network，GCN）是一种基于图数据的神经网络模型，它能够捕捉图中的结构和特征信息。GCN由于其简单的设计和强大的性能，已经在多个领域得到了广泛的应用，例如交通预测、蛋白质性质预测、推荐系统等。因此，提高GCN的计算速度对于促进相关领域的发展和改善人们生活质量具有重要的价值。

在简单的GCN中，我们有节点特征（代表节点的数据，通常是向量表示）和图的结构（表示节点图和连接）。卷积的想法来自于图像（image），之后引进到图中。



GCN直接在图上工作，利用图的结构信息，其主要思想如下：对于每个节点，我们从它的所有邻居节点处获取其特征信息（当然包括自身信息）；最后将每个顶点的信息输入到神经网络中。重复上述操作，我们就得到了多层的图神经网络。



多层图神经网络

一层的GCN可以用以下公式来表述：

$$X^{(l+1)} = \alpha(\hat{A}X^{(l)}W^{(l)})$$

其中l表示层号，α（$ReLU$、$SIGMOID$等）表示激活函数，$\hat{A} \in R^{|V|\times|V|}$为归一化的图邻接矩阵；$X^{(l)} \in R^{|V|\times F_l}$是输入的顶点特征矩阵（$X^{(0)}$为初始的节点特征）；$W^{(l)} \in R^{F_l \times F_{(l+1)}}$为神经网络的权重矩阵；$X^{(l+1)} \in R^{|V|\times F_{l+1}}$为输出的顶点特征矩阵。其中，归一化的邻接矩阵$\hat{A}$的计算方式为：$\hat{A} = D^{-1/2}AD^{1/2}$。

在这里A为图邻接矩阵（数据集文件中默认包含自环），D为对应的度矩阵（是一个对角矩阵）

2 题目介绍

公式：

本项目中的GCN由两个图卷积层构成，第一层的激活函数使用$ReLU$，第二层使用$LogSoftmax$；

$ReLU$函数定义为：$ReLU(x) = max(0, x)$;

$LogSoftmax$函数定义为：$LogSoftmax(X_{i,j}^{(l)}) = (X_{i,j}^{(l)} - X_{i,max}^{(l)}) - \log\left(\sum_{c=0}^{F_l-1} e^{X_{i,c}^{(l)} - X_{(i,max)}^{l}}\right)$

，$X_{i,max}^{l} = max\left(X_{(i,0)}^{(l)}, \ldots, X_{i,F_l-1}^{l}\right)$

综上所述，本文使用的唯一公式如下：

$$Z = f(X, A) = LogSoftmax\left(\hat{A} \cdot ReLU\left(\hat{A}XW^{(0)}\right) \cdot W^{(1)}\right)$$

文件介绍：

在本项目的数据集文件如下：./graph/*.txt；./embedding/ *.txt是每个图文件对应的特征（feature）矩阵；./weight/ *.bin为分别为两层GCN的参数矩阵。

- 图结构文件为文本文件，第一行两个整数分别为图顶点数量（v_num）和边数量，之后每一行为一条边，格式为“源顶点id 目的顶点id”，顶点id从0开始
- 图结构文件中包含自环（即有边“i i”），包含反向边（即同时有边“i j”和边“j i”）

- 输入顶点特征矩阵文件为二进制文件，包含$v_num * F_0$个float32，大小为$v_num * F_0$*4字节
- 第一层权重矩阵文件为二进制文件，包含$F0 * F1$个float32，大小为$F0 * F1 * 4$字节
- 第二层权重矩阵文件为二进制文件，包含$F1 * F2$个float32，大小为$F1 * F2 * 4$字节

./example/gcn.cpp 是一个示例代码，其中包括读文件、预处理、计算、输出结果等。**其中readGraph()函数不可修改**

- 读取文件、分配内存、和数组初始化置0的时间均不统计在执行时间内
- 预处理时间（例如顶点排序）等须计入执行时间

./example/makefile 是make文件，可以修改，也可以使用cmake工具编译代码

./run.sh 是运行代码的例子

- 可执行程序需接收7个参数，分别为：输入顶点特征长度F_0，第一层顶点特征长度F_1，第二次顶点特征长度F_2，图结构文件名，输入顶点特征矩阵文件名，

  第一层权重矩阵文件名，第二层权重矩阵文件名，在run.sh中有例子
- 可执行程序需输出两个值，分别为：最大的顶点特征矩阵行和执行时间
- 如果使用第三方库需要提供具体说明

本项目是可以直接运行的，同学们拿到代码之后可以运行基础代码得到相应的结果，优化代码要求结果准确。

3 优化目标

3.1 基础优化：优化代码的运行

首先，代码使用了两个矩阵相乘操作分别为XW、AX，即feature矩阵乘以参数矩阵的NN操作以及邻接矩阵乘以feature的图传播操作，其中X和W是两个稠密的矩阵，而A是一个稀疏矩阵。稠密矩阵的乘法操作可以有一些优化的方法，包括但不限于向量化加速、多线程处理、优化数据读取等等。

其次，预处理阶段是将row_graph转化为邻接矩阵A。稀疏矩阵乘法AX比较浪费时间，因为A中大部分元素都是0，因此可以从这个矩阵的实际运行方式入手，考虑矩阵的压缩，不处理0元素。包括但不限于以上方法。

3.2 进阶优化：优化数据结构以及其他算子

首先，使用邻接矩阵或者压缩邻接矩阵存储的图数据，在稀疏矩阵相乘阶段有比较低的性能。

其次，$ReLU$函数、$LogSoftmax$函数可以使用向量化等手段优化。

最后，还会有一些算子融合的可能。比如某两个操作是可以在同一个循环中实现。

3.3 终极优化：多进程或者分布式处理以及GPU加速

- 现实场景中会有一些比较大的数据集，处理这些数据集的时候单机环境可能放不下相应的feature矩阵，甚至放不下图数据，这就需要使用分布式技术处理（如果分布式比较困难，可以使用多进程模拟分布式环境的执行，只使用小一些的数据集）
  - 在分布式场景下要注意图划分（每个node存哪些节点，尽量较少跨node的边）、负载均衡（每个node处理相同的任务量）等问题
- 或者使用GPU加速，矩阵乘法、稀疏矩阵乘法都有相应的比较成熟的库来加速

3.4 优化说明

基础优化以及进阶优化建议尽量完成，终极优化是加分项。

4 提交说明

4.1 提交材料：

- 程序源代码：包含完整目录结构的源代码，其中包含makefile文件
- 执行makefile文件应编译出可执行文件（如需安装第三方库，请提供具体说明）
- 技术报告文档，包括但不限于算法介绍、设计思路及方法、实验分析，还有从项目中学到了什么

4.2 提交地址：

创建自己的Github账户，并将相关文件上传到Github账户中（需要公开），最后将Github地址提交。

Github项目文件结构如下：

```
your_name-iDC-Mlsys_interview
├── your_name
|       ├── source_code.cpp
|       ├── makefile
|       └── README
├── your_name_report
|       └── report.pdf
└── your_name.exe
```