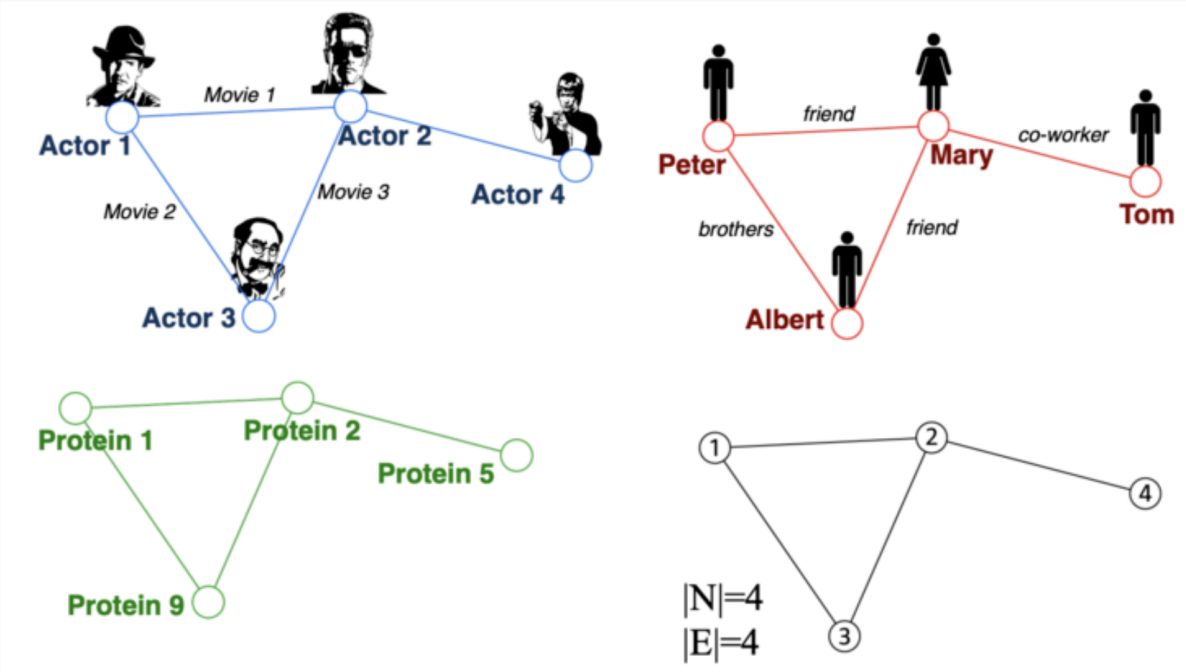
题目 图神经网络计算的优化

1 背景

1.1 图

图（Graph）是一种普遍存在的数据结构，并被广泛用于处理真实世界中的问题，例如社交网络、交通网络、金融系统等。

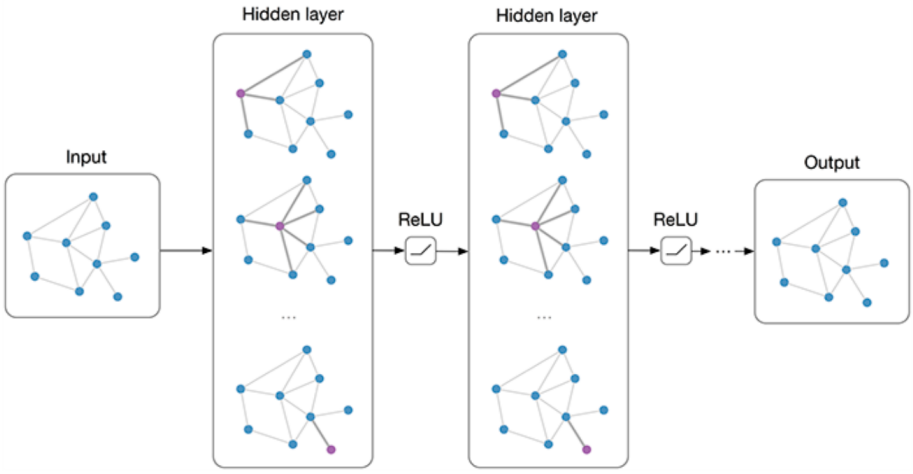


1.2 图卷积神经网络

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，数据类型主要有两种，节点特征（通常是多维向量）和图结构（顶点与边）。

在GCN模型的一层训练中，每个顶点会聚合其邻居顶点的特征信息并与自己的信息相结合，得到聚合结果后，再将其输入到神经网络中。重复这个过程，可以得到更深层的GCN模型。



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 表示模型层号， α （ $ReLU$ 、 $Sigmoid$ 等）表示激活函数， $\hat{A}$ 为归一化的图邻接矩阵； $X^{(l)}$ 是输入的顶点特征矩阵（ $X^{(0)}$ 为初始的节点特征）； $W^{(l)}$ 为神经网络的权重矩阵； $X^{(l+1)}$ 为输出的顶点特征矩阵。

2 题目介绍

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。

2.1 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：

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

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

综上所述，本文使用的唯一公式如下：

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

2.2 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 1：优化稠密矩阵相乘（改进 `XW()`）

X 和 W 是两个稠密的矩阵。稠密矩阵的乘法操作的优化方法包括但不限于向量化加速、多线程处理（`OpenMP`）、计算模式等等。

3.2 基础优化 2：优化稀疏矩阵相乘（改进 `AX()`）

首先， A 是一个稀疏矩阵，因为 A 中大部分元素都是0。稀疏矩阵乘法乘法操作的优化方法包括但不限于向量化加速、多线程处理、矩阵压缩等等。

另外，代码示例中使用是邻接表格式，在实现稀疏矩阵相乘阶段(AX)有较低的性能。

CSR是一种更好的稀疏矩阵存储方式，推荐在预处理阶段将数据格式转为CSR，并在实现稀疏矩阵相乘阶段(AX)使用CSR。

3.3 基础优化 3：其他函数优化（选做）

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

3.4 进阶优化 1：GPU加速（选做）

使用GPU加速矩阵乘法（可以使用外部库）。

3.5 进阶优化 2：多进程或者分布式处理（选做）

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（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）。
任务目标：实现分布式环境下的GCN数据并行训练。

3.6 优化说明

基础优化1和2至少完成一个，选做部分是加分项，也可以使用文档中未提及的加速方法，也是加分项。

4 验收材料

程序源代码。
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。