Assignment 2 Report

Xie Zhiyao:2009853G-II20-0195

实验结果：

共完成了三个版本的SGD计算，最开始先完成了一个base\_sgd，是使用numpy中的dot方法，直接取出矩阵中的整行来进行计算：文本

描述已自动生成

运行结果：

图形用户界面, 文本

描述已自动生成

可以观察到在第十三次迭代的时候RMSE已经达到了0.9，后续收敛速度开始降低，一轮迭代的耗时为11毫秒左右。在这个循环的计算中，Time是遍历了整个稀疏矩阵的结果，于是想到在排除掉稀疏矩阵中的nan后再进行迭代运行所消耗的时间会更快，于是完成了在非并行情况下，只使用numpy的第二个版本的SGD计算（unparallel\_fastest\_sgd）:

文本

描述已自动生成

其中：

文本

描述已自动生成

为筛选出矩阵中非nan的索引进行遍历，运行结果为：

文本

描述已自动生成

仅仅在第四轮迭代中RMSE就已经收敛到0.9，且Time中的用时不到5毫秒。

以上结果得益于numpy中对运行速度的优化。

随后根据题目要求做了分块和并行的版本（block\_parallel\_sgd）,首先计算了在不使用numpy的情况下，进行三层循环，只用了一条进程，没有使用分块的情况下，耗时非常漫长没能运行出最后的结果，因此进行检查。计算了在这个实验中进行一次进程传输时需要传递的数据量（主要为P，Q，R矩阵中的数据量）：

另外在t\_process\_pipe程序中模拟了进行一次数据传输至进程再取回所需要的时长：

文本

描述已自动生成

发现往进程中传送数据的耗时较长：

文本, 应用程序

描述已自动生成

可见在进程中使用.get()方法取出计算好的数据时，除了第一次取数据有一个245毫秒左右的预热过程外，每次取数据也有35毫秒左右的耗时，相比于进程中对数据的计算速度，主进程与子进程之间的数据传输速度过于漫长。考虑到这个问题，查阅资料后发现应该可以用共享内存的方式解决。然而根据作业中的要求，首先考虑分块的方式来减少主进程与子进程之间需要的传输次数来减少总耗时。

在分块的设计中，首先为P，Q矩阵进行块划分，其中根据需要的进程数进行动态划分，例如我的电脑是四核心，因此划分为4个进程，矩阵划分为4\*4=16块，当然也可以划分为5个进程共5\*5=25块，经过实验后发现4个进程时效果最好。

分块设计即意味着分批次将互不影响的数据块喂入一个子进程，逻辑如下：

白板上写着字

描述已自动生成

也就是说第一个进程喂进去的是序号为（1，6，11，16）的数据块，第二个进程喂进去的是（2，7，12，13）的数据块，以此类推，因此需要计算每次喂进子进程中的块排序，算法设计如下：

文本

描述已自动生成

返回值为块序号。在后续的算法中会根据块序号去取矩阵中相应的行坐标和列坐标，以此找到需要传输到进程中的数据块。因此divBlock\_matrix中返回根据线程数划分的相应的行坐标和列坐标。

文本

描述已自动生成

主进程中取数据块的方式则如下：

文本

描述已自动生成

分块+多进程的运行结果为：

文本

描述已自动生成

可以看到RMSE的结果并不如numpy中可以快速收敛到0.9以下，以及将四个进程中.get()的耗时加起来，可以发现是与总耗时相差无几的（2.3+1.9+1.5+1.5=7.2），总耗时7.6，可见大部分的时间都消耗在了数据的传输上。在时间有限的情况下最终没有进行share memory的优化，但在后面的课程中感受到这应该是一个可行的方案，如果进程之间有一块共享数据的内存，那么主进程就不需要对子进程进程数据的反复读取，而是在第一个子进程计算完以后，将计算结果共享给下一个子进程，主进程只进行最初和最后的读取，将会大大减少一次迭代所需要的时间。

补充部分：

1. 数据处理：在dataPre中，将数据按8:2划分为训练数据和测试数据，训练数据用于训练得出收敛后的隐因子矩阵P，Q，测试数据用于在经过迭代收敛后得到的P，Q矩阵中计算RMSE，以及label矩阵R，统一整理为DataFrame格式返回：

文本

描述已自动生成

1. 初始化参数：根据作业要求中的公式计算初始化的偏置Mu, b1和b2，以及P，Q矩阵的初始化参数也设置了一定范围内的随机值：

文本

描述已自动生成

文本

描述已自动生成

在实验过程中发现初始化参数的设置有助于训练过程的快速收敛。例如：在P，Q矩阵一开始随机赋值的时候，迭代至100次以上才勉强收敛到0.9，而在限制了初始值的范围后，20次以内的迭代就能收敛到0.9以下。