基于 MPI 和 OpenMP 混合编程的推荐系统混合模型

王锡淮 16337236

目录

1 项目介绍

本项目是一个基于用户行为模型和用户评分模型的混合模型的推荐系统，其核心算法是基于物品的协同过滤算法，实现方法依靠的是 MPI 和 OpenMP。项目训练使用的是 MoviesLen 数据集，该推荐系统能够向指定用户推荐指定数目的电影。项目主页是https://github.com/Leo-xh/Recommendation-System-Based-on-MPI-OpenMP.

2 推荐系统设计与分析

2.1 基于物品的协同过滤算法

基于物品的协同过滤算法主要分为两步：

1. 计算物品之间的相似度。
2. 根据物品的相似度和用户的历史行为给用户生成推荐列表。

其中物品相似度的归一化，如公式1所示，能够增加推荐的准确度，提高推荐的覆盖率和多样性。

$$w'_{ij} = \frac{w_{ij}}{\max_j w_{ij}} \tag{1}$$

基于物品的协同过滤算法着重于维系用户的历史兴趣，其推荐更加个性化，反映了用户自己的兴趣传承，同时还能够顾及长尾物品和对用户给出推荐解释。

2.2 基于用户行为的推荐系统

在这个方法中，物品之间相似度的计算方法如公式2所示，该公式考虑了活跃用户对物品相似度的贡献应该小于不活跃的用户。

$$w_{ij} = \frac{\sum_{u \in N(i) \cap N(j)} \frac{1}{log(1+|V(u)|)}}{\sqrt{|N(i)||N(j)|}} \tag{2}$$

其中 $N(i)$ 指的是对物品 i 有过行为（评价）的用户，$N(j)$ 指的是对物品 j 有过行为的用户。

而该方法中用户 u 对物品 i 的兴趣度量公式3如所示，其中协同过滤体现在其考虑了用户 u 对与物品 i 最相似的 k 的物品的评价。

$$p_{ij} = \sum_{i \in N(u) \cap S(j,k)} w_{ij} r_{ui} \tag{3}$$

其中 p_{ij} 是用户 u 对物品 i 的兴趣，$N(u)$ 是用户 u 喜欢的物品的集合，$S(j, K)$ 是和物品 j 最相似的 K 个物品的集合，w_{ji} 是物品 j 和 i 的相似度，r_{ui} 是用户 u 对物品 i 的兴趣，在基于用户行为的模型中有过动作为 1，否则为 0。

2.2.1 并行性分析

由简单的数据相关图可知该模型中有许多可以并行化的地方，主要有两个，一是相似度矩阵的计算，二是兴趣矩阵的计算，都可以使用 MPI 来进行进程间的并行化或者 OpenMP 来进行线程间的并行化。详细设计见章节3。

2.3 基于用户评分的推荐系统

在设计了基于用户行为的推荐系统后，萌生使用用户的评分信息来预测用户对物品的评分，进而向用户推荐评分最高的物品的想法，而且这样也能进一步利用数据集的信息。

这个模型仍然分为计算物品之间的相似度和根据物品的相似度和用户的历史行为给用户生成推荐列表两步，只是相似度计算方式改变，以及生成推荐列表的方式改为根据对于用户对物品评分的预测。

物品相似度计算公式如公式所示，这是修正过的余弦相似度。

$$w_{ij} = \frac{\sum_{u \in U} (r_{ui} - \bar{r}_u) \times (r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{ui} - \bar{r}_u)^2 \sum_{u \in U} (r_{uj} - \bar{r}_u)^2}} \tag{4}$$

其中 $\bar{r}_u$ 是用户 u 给出的所有评分的平均数，U 是用户的集合，r_{ui} 和 r_{uj} 则是用户 u 对物品 i 和 j 的评分。

而用户对物品的评分预测的估计公式如公式5所示，

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in S(i,K) \cap N(u)} w_{ij} (r_{uj} - \bar{r}_i)}{\sum_{j \in S(i,K) \cap N(u)} |w_{ij}|} \tag{5}$$

其中 $\hat{r}_{ui}$ 代表用户 u 对物品 i 的评分的预测值，r_{ij} 为用户对物品的评分，其余各项与前述一致。

2.3.1 并行性分析

由简单的数据相关图可知，与前面的基于用户行为的模型相对比，这一模型的计算量明显增多，而且并行性也更多，比如用户评分的均值和物品收到的评分的均值可以使用结点单独计算，计算量的增加也可以隐藏线程创建和共享变量的处理时间，于是可以使用更多的线程。详细设计见章节3。

2.4 模型混合

现在我们有了两个不同的预测器，且两个预测器都可以计算出类似的值，一个是用户对物体的兴趣，另一个是用户对物品的评分，其实这两者是一致的，于是一个自然的想法是将两者融合起来获得最低的预测误差。

本实验中使用的是简单的线性融合，最终的预测器 $\hat{r}$ 是这两个预测器的线性加权，如公式6所示。

$$\hat{r} = \sum_{k=1}^{2} \alpha_k \hat{r}^{(k)} \tag{6}$$

其困难在于权重 α_k 的确定，这是一个回归问题，本实验中使用最小二乘法。

2.4.1 最小二乘法的应用

1. 假设数据集已经被分为了训练集 A 和测试集 B，那么首先需要将训练集 A 按照相同的分割方法分为 A1 和 A2，其中 A2 的生成方法和 B 的生成方法一致，且大小相似。

2. 在 A1 上训练 K 个不同的预测器，在 A2 上作出预测。因为我们知道 A2 上的真实评分值，所以可以在 A2 上利用最小二乘法计算出线性融合系数 α_k。

3. 在 A 上训练 K 个不同的预测器，在 B 上作出预测，并且将这 K 个预测器在 B 上的预测结果按照已经得到的线性融合系数加权融合，以得到最终的预测结果。

3 实现

下面根据 Foster 的并行程序设计的方法论：划分-通信-聚集-映射四个环节来设计上述模型基于 MPI 和 OpenMP 的并行实现，并详细叙述了部署过程。

3.1 MPI 程序的设计

3.1.1 划分

从最大化原始任务的数目的目的出发，在两个模型中都可以将相似度矩阵划分成物品数目 × 物品数目的原始任务，每个任务是一个相似度，而且在基于用户评分的模型中计算用户评分的平均数和计算物品被评分的平均数这两个过程还可以独立称为任务，但是由于实现时数据级的存储方式，用户的评分平均数计算较为简单，速度较快，将其独立作为任务的通信代价和等待代价大于得到的收益，于是只将物品得分平均数的计算作为任务独立出来。用户对物品的兴趣（预测的评分）矩阵计算量较小，所以不做划分。

3.1.2 通信

本实验中使用的是局部通信的方式，将一个结点作为 master 结点，其任务是收集其余结点的计算结果，根据这些数据计算用户对于物品的兴趣（预测的评分），其余结点分别计算物品得分平均值（基于用户评分的模型中使用）和物品相似度。

3.1.3 聚集

观察公式2和公式4可知按行聚集上述原始任务能够得到较高效率，提高了并行算法的局部性，复制的信息也较少，而且具有拓展性，当节点数增多时也能够较好的适应。

负载均衡。由于相似度矩阵关于主轴对称，直接按照行来聚集任务显然做了许多不必要的重复工作，于是可以只计算下三角矩阵，这就引出了各结点之间的负载均衡问题，理想的聚集结果应该如图所示，图中的三角形和梯形面积应当相等。

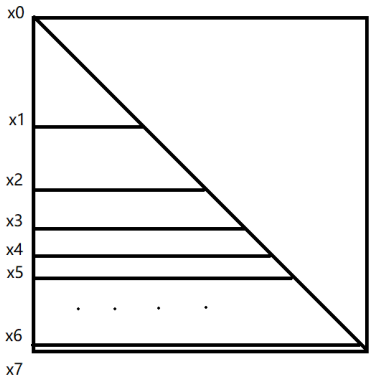

图 1: 理想的负载均衡结果

其中 $x_i, i = 0, 1, 2, \ldots, N+1$，N 为聚集出来的任务数，要使矩阵得到准确的划分，可以用上下两个几何体的面积相等关系求得，即公式7所示。

$$\begin{aligned} &\frac{x_1^2}{2} = L/N \\ &x_2 = \sqrt[2]{x_1} \\ &x_{k+1} = \sqrt[2]{2x_k^2 - x_{k-1}^2},\ k = 2, 3, \ldots, N-1 \end{aligned} \tag{7}$$

其中 L 为原始任务数目。

3.1.4 映射

因为已经考虑了负载均衡问题，直接使用静态分配任务的方式，将一个聚集好的任务映射给一个处理器即可，这样能充分利用处理器和最小化处理器之间的通信。

3.2 OpenMP 的应用

首先根据 MPI 程序轮廓刻画，如表3.2所示，该表展示了在 6 个进程上执行的基于用户行为的模型的结果，存储数据部分不能优化，可见计算相似度矩阵需要使用多线程进行优化。

过程	时间
计算相似度矩阵	12.53s
计算兴趣矩阵	2.74s
存储数据	33.46s

在上面也提到过，由于相似度矩阵巨大，用户对物品的兴趣（预测的评分）的计算依赖于该矩阵，如果通过 MPI 将这一过程并行化需要较大的通信代价，对于内存的占用也较多，因此在 master 结点对其使用 OpenMp 进行优化。

此外，从图中也可以看出，在各个结点中，相似度的计算也是该程序的时间瓶颈，但是计算量相对较小，于是可以用较少的线程进行优化加速。

3.3 模型混合

模型混合部分的主要内容是将训练集（关于训练集和测试集的划分见章节4.2部分）按照划分训练集和测试集的方式划分成训练集和验证集，再按照验证集中含有的用户对物品的兴趣（评分）进行对应的预测，然后使用最小二乘法进行曲线拟合。

在实现的时候，由于高斯消元法的时间复杂度为 $O(n^3)$，完成的实现版本效率也较低，计算需要大量时间，于是使用 GNU Scientific Library 来进行求解。

3.4 部署

要将 MPI 程序部署在多台计算机上实现分布式计算，可以利用 mpirun 或者 mpiexec 程序的可选参数指定计算机的 IP 地址。然而要使用这个参数需要解决的是计算机本身的互联问题，首先是 ssh 的配置，最好将多台计算机配置成无需密码即可互联，在 windows 下操作较为复杂，还需要修改防火墙的入站规则等等，建议在 linux 下使用，而且在 linux 下的 OpenMPI 也较为方便，详细方法参考https://blog.csdn.net/bendanban/article/details/40710217。除了配置计算机互联，还需要注意的是在多台主机上建立相同的用户名，将数据文件放在相同的路径上，以及安装相同的 MPI 版本等等。

4 结果

首先是使用 split 程序将数据集按照 8:1 随机均匀划分成训练集和测试集，其次使用 parallelA 和 parallelR 程序计算用户对物品的兴趣和评分，并将相似度矩阵和兴趣值矩阵保存下来。

该项目最后得到的程序是一个 predict 程序，其根据之前所计算出来的用户对物品的兴趣或评分排序，输出前 K 个推荐物品。用该程序预测验证集中的评分，进行模型混合，得到混合模型。

4.1 结果展示

4.1.1 训练过程

```
mpi@Coxy-Ubuntu:~$ mpirun --hostfile host ./parallelA 10 2
Rank 3 in processer ht-50 is on
Rank 4 in processer ht-50 is on
Rank 5 in processer ht-50 is on
Rank 2 in processer ht-50 is on
Rank 1 in processer Coxy-Ubuntu is on
Rank 0 in processer Coxy-Ubuntu is on
memory fine...
calculating weights...
        distributing tasks... 5 nodes and 1825 items per node.
        collecting data from node 1 ...
Node 5 calculating weights... 3.704096 s
Node 4 calculating weights... 4.401211 s
Node 3 calculating weights... 4.896744 s
Node 2 calculating weights... 8.489861 s
Node 1 calculating weights... 17.160913 s
node 1 in Coxy-Ubuntu sending weights to master...
        collecting data from node 2 ...
```

但是在训练过程中出现主机收不到信息的情况，原因未知。

4.1.2 使用结果

使用方法是:

```
Usage: executable [NumberOfPrediction] [UserId]
```

一个示例如下所示，显示的是对用户 8 推荐的 10 个电影。

```
./predict 10 8
1: 5x2 (2004)
```

2: Night and the City (1950)
3: Night Watch (Nochnoy dozor) (2004)
4: Kung Fu Hustle (Gong fu) (2004)
5: Zelary (2003)
6: ”Seven-Per-Cent Solution, The (1976)”
7: ”Three Musketeers, The (1973)”
8: Masculin Féminin (1966)
9: Control (Kontroll) (2003)
10: The Boyfriend School (1990)

4.2 评测指标

选取评测指标为评分的均方根误差、TopN 推荐的准确率和召回率，以及覆盖率。

4.3 评测结果

混合模型的评测结果如表4.3所示，

指标	结果
准确率	23.58%
召回率	13.40%
覆盖率	29.11%
均方根误差	0.73

5 总结

本次项目中我阅读了两本书籍,《推荐系统实践》和《Parallel Programming in C with MPI and OpenMP》，前者讲述的是如何从零构造起一个推荐系统，并且告诉我什么叫一个好的推荐系统与各种指标意义及其评测方法，后者则教会了我设计并行分布式程序的设计方法论以及 OpenMP 在 MPI 程序中起到什么作用，感触最深的是《推荐系统实践》这本书把现在我遇到推荐系统，比如亚马逊和豆瓣等，的原理一层层剖析开来；还有 Foster 方法论：划

分-通信-聚集-映射，也让我设计程序有章法可循；实现一个 c++STL 通用最小二乘法着实不易，而且效率太低，对于本项目中的大型矩阵无能为力，之后我发现了 C/C++ 的数值计算库，又一次发现 windows 下 C 和 C++ 的支持不好，于是在 linux 下使用 GNU Scientific Library 求解；还有就是配置多主机并行是在花费了太多时间，原因是一开始在 windows 上配置，缺少文献，加上系统本身不太支持，让我明白开发分布式程序还是要在 linux 上进行。

当然，通过这个项目，我也明白了这多处理器级别并行的两种方式，MPI 的多主机方式和 OpenMP 的处理器多核方式，对于优化程序的互补作用，无论是在数据级并行还是功能级并行都可以用到。

除此之外，在《推荐系统实践》一书中也有基于机器学习方法的推荐系统算法，通过对这些算法的思考，我也明白了设计并行和分布式程序和设计机器学习算法其实并不冲突，分布式程序的设计注重于实现方式的设计，而机器学习算法的设计注重于实现目标，二者研究于不同的方面。

本项目的缺点在于算法过于简单，甚至没有考虑物品流行度的问题，同时算法简单但是计算量大，基于用户评分的模型需要在两台电脑上训练 4 个小时。同时也没有解决训练过程中可能出现的错误问题。

A References

参考文献

[1] 项亮, 推荐系统实践, 人民邮电出版社.

[2] Michael.J.Quinn, *Parallel Programming in C with MPI and OpenMP*, 清华大学出版社.

[3] bendandan, https://blog.csdn.net/bendanban/article/details/40710217, 无需超级用户 *mpi* 多机执行.