计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(5):166−170 [doi: 10.15888/j.cnki.csa.006351]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Hadoop 的协同过滤并行化算法①

曹　霞, 谢颖华

(东华大学 信息科学与技术学院, 上海 201620)
通讯作者: 曹　霞, E-mail: caoxia199323@163.com

摘　要: 在针对大数据的迅速增长, 为了改善协同过滤算法的推荐效率, 使得推荐精度越来越高, 提出基于 Hadoop 平台的协同过滤并行化算法, 将传统的基于用户的协同过滤在 Hadoop 平台下进行 MapReduce 编程模型, 实现并行化. 通过利用 MovieLens 公用数据集对改进前后的算法对比, 验证了并行化的协同过滤效率更高, 也更加适合大规模数据的推荐.

关键词: 协同过滤; Hadoop; 并行化; MapReduce

引用格式: 曹霞,谢颖华.基于 Hadoop 的协同过滤并行化算法.计算机系统应用,2018,27(5):166–170. http://www.c-s-a.org.cn/1003-3254/6351.html

Parallel Algorithm of Collaborative Filtering Based on Hadoop

CAO Xia, XIE Ying-Hua

(College of Information Science and Technology, Donghua University, Shanghai 201620, China)

Abstract: In order to improve the recommendation efficiency of collaborative filtering algorithm, this study proposes a collaborative filtering parallelization algorithm based on Hadoop platform. The traditional user-based collaborative filtering is carried out under Hadoop platform for MapReduce Programming model, to achieve parallelization. By using the MovieLens common data set to improve the comparison before and after the algorithm, verify that the parallel collaborative filtering efficiency is higher, and also more suitable for large-scale data recommendation.

Key words: collaborative filtering; Hadoop; clustering analysis; MapReduce

1　引言

随着互联网的快速发展, 推荐系统越来越受到大家的关注, 很多人都开始研究如何优化推荐系统. 更有很多电子商务平台也在开始使用推荐系统, 比如说 Amazon、淘宝、京东, 以及各种我们现在使用的手机软件 APP, 随处可见的推荐系统.

协同过滤算法主要分为两种类型, 一种是基于项目 (item) 的协同过滤, 一种是基于用户 (user) 的协同过滤[1]. 如今这两种已经不能满足各类网站的推荐效率, 有很多人陆续提出了对协同过滤算法的改进方法. 如有人提出基于聚类模型的协同过滤算法[2], 原理就是用聚类对 user 或 item 分组, 然后用协同过滤算法是对一件 item 评分, 将评分最高的 item 推荐给 user. 还有人提出了基于用户的 Hadoop 协同过滤算法[3], 也就是本文将进行详细描述的思想. 更有人提出的基于 ALS (Alternating Least Squares) 的协同过滤[4], ALS 算法是基于矩阵分解的协同过滤算法中的一种, 在 Soven 的 oryx 框架中, 推荐算法便是采用的这种算法.

随着海量信息的暴增, 数据没有办法能够很好地储存和计算, 这个时候就需要我们对推荐算法做一些改进. 不光对算法本身做出优化, 也要在运行平台上做出很大的变化, 这时我们就加入了 Hadoop 平台. 正如我们所知道的, 其实一直以来我们的协同过滤算法都是以单机的形式来运行的, 这在某种程度上就会影响

① 收稿时间: 2017-09-01; 修改时间: 2017-09-20; 采用时间: 2017-09-29; csa 在线出版时间: 2018-04-23

运行速率, 随着数据的增多, 也越来越不能满足如此大量数据的运算. 因此本文提出基于 Hadoop 这样的云平台来实现基于用户的协同过滤的并行化. 因此多机形式下运行推荐算法成为推荐算法研究中一个新的研究方向.

本文将传统的协同过滤在 Hadoop 平台下实现并行化, 也就是基于 MapReduce 的推荐过程, 各个部分功能分模块开发, 互不影响, 从而方便地进行推荐算法的扩展.

2 相关概念

2.1 关于协同过滤

协同过滤算法的推荐主要有 5 步, 第一步为根据原理建立评分模型, 第二步为选出可作为模型中的中间标准项目, 第三步是根据公式计算中间项目的最近邻居项目集, 第四步是从最近项目集选出最优解, 第五步是推荐完成. 本文采用普遍应用的统计学的准确性度量——平均绝对误差 (MAE) 以及单机运行时间和 Hadoop 集群运行时间的加速比.

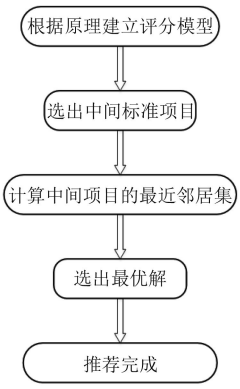

图 1 协同过滤推荐算法流程图

上面已经提到过, 协同过滤算法的推荐过程分以上 5 个过程这五步当中最重要的就是第三步, 也就是最近邻居集的计算. 一般常见的相似性度量方法有三种: 皮尔逊 (Pearson) 相关相似性、余弦相似性、修正的余弦相似性等, 下面依次介绍这 3 种度量方法.

(1) 皮尔逊相关相似性

皮尔逊相关相似性是度量两个变量的线性相关性的方法, 计算公式[5]如下:

$$s(n,m)=\frac{\sum_{p\in I_{n,m}}\left(r_{n,p}-\overline{r_i}\right)\left(r_{m,p}-\overline{r_m}\right)}{\sqrt{\sum_{p\in I_{n,m}}\left(r_{n,p}-\overline{r_i}\right)^2}\sqrt{\sum_{p\in I_{m,p}}\left(r_{m,p}-\overline{r_m}\right)^2}} \quad (1)$$

其中$I_{n,m}$表示用户 m 和用户 n 共同评分的项目集合, $r_{n,p}$和$r_{m,p}$分别表示用户 n 和用户 m 对项目 p 的评分; $\overline{r_n}$和$\overline{r_m}$则分别代表用户 n 和用户 m 对系统中所有项目的评分均值.

(2) 余弦相似性

在协同过滤中, 用户的评分向量是用户的兴趣描述, 通过计算两用户的评分向量之间的夹角余弦来度量用户的相似度. 计算公式[5]如下:

$$s(n,m)=\frac{\sum_{p=1}^{n_i} r_{n,p}r_{m,p}}{\sqrt{\sum_{p=1}^{n_i} r_{n,p}{}^2}\sqrt{\sum_{p=1}^{n_i} r_{m,p}{}^2}} \quad (2)$$

(3) 修正的余弦相似性

在实际的推荐系统中, 对同一项目, 用户的评分尺度可能不同. 修正的余弦相似度将该问题考虑在内, 通过用户的评分均值来消除评分尺度的不同. 计算公式[5]如下:

$$s(n,m)=\frac{\sum_{p=1}^{n_i}\left(r_{n,p}-\overline{r_n}\right)\left(r_{m,p}-\overline{r_m}\right)}{\sqrt{\sum_{p=1}^{n_i}\left(r_{n,p}-\overline{r_n}\right)^2}\sqrt{\sum_{p=1}^{n_i}\left(r_{m,p}-r_m\right)^2}} \quad (3)$$

2.2 关于 MapReduce 编程模型

Hadoop 是 Apache 软件基金会旗下的一个开源分布式计算平台, 以 HDFS 和 MapReduce 为两大核心. MapReduce 是一种编程模型, 用于大规模数据集 (大于 1 TB) 的并行运算. 当前的软件实现是指定一个 Map(映射) 函数, 用来把一组键值对映射成一组新的键值对, 指定并发的 Reduce(归约) 函数, 用来保证所有映射的键值对中的每一个共享相同的键组.

一个 Map/Reduce 作业 (job) 通常会把输入的数据集切分为若干独立的数据块, 由 Map 任务 (task) 以完全并行的方式处理它们. 框架会对 Map 的输出先进行排序, 然后把结果输入给 Reduce 任务. 通常作业的输入和输出都会被存储在文件系统中. 整个框架负责任务的调度和监控, 以及重新执行已经失败的任务.

通常, MapReduce 框架和分布式文件系统是运行在一组相同的节点上的, 也就是说, 计算节点和存储节点通常在一起. MapReduce 框架由一个单独的 master JobTracker 和每个集群节点一个 slave TaskTracker 共同组成. master 负责调度构成一个作业的所有任务, 这些任务分布在不同的 slave 上, master 监控它们的执行, 重新执行已经失败的任务. 而 slave 仅负责执行由 master 指派的任务. 应用程序至少应该指明输入/输出的位置 (路径), 并通过实现合适的接口或抽象类提供 map 和 reduce 函数. 再加上其他作业的参数, 就构成了作业配

置 (job configuration). 然后, Hadoop 的 job client 提交作业 (jar 包/可执行程序等) 和配置信息给 JobTracker, 后者负责分发这些软件和配置信息给 slave、调度任务并监控它们的执行, 同时提供状态和诊断信息给 job client.

3 基于 Hadoop 的协同过滤算法

3.1 算法原理

Hadoop 是一个开源的分布式计算开源框架, 它提供了分布式文件系统 (Hadoop Distributed File System, HDFS) 和支持 MapReduce 分布式计算的软件架构. MapReduce 框架的计算过程分为 Map 和 Reduce 两个阶段. 在 MapReduce 作业中, 输入的数据集被切分为若干独立的数据块, 先由 Map 任务并行处理, 对 Map 的输出数据排序后, 再作为输入数据交由 Reduce 任务处理, 最终输出计算结果. 每个阶段的输入输出数据格式是<key, value>形式的键值对. 开发者只需编写 Map 和 Reduce 阶段的映射函数, 任务装载、调度和节点间的通信由 Hadoop 自动完成.

基于 MapReduce 的协同过滤, 这时就需要我们输入用户—项目评分测试集 M, 项目聚类集合 C 以及聚类中心 c_0, 最后得出推荐结果. 用户-项目评分矩阵如下图, 假设有 n 个项目和 m 个用户, r_{ij} 就是用户 j 对第 i 个项目的评分, 以此类推[6].

表 1 用户-项目评分矩阵

	项目1	项目2	…	项目i	…	项目n
用户1	r_{11}	r_{12}		r_{1i}		r_{1n}
用户2	r_{21}	r_{22}		r_{2i}		r_{2n}
…						
用户j	r_{j1}	r_{j2}		r_{ji}		r_{jn}
…						
用户m	r_{m1}	r_{m2}		r_{mi}		r_{mn}

步骤如下:

(1) 根据上面所提到的公式, 我们将矩阵中要计算的项目与我们所选好的目标项目 c_0 进行计算得到 s 结果.

(2) 将计算出来的 s 与我们设定好的相似性阈值比较, 把相似度大于等于相似性阈值的项目放到一个集合 W 中.

(3) 在 W 中进行我们的最后一步, 基于 Hadoop 平台上的 MapReduce 分块计算. 系统自动将一个作业待处理的大数据划分为很多个数据块, 每个数据块对应于一个计算任务, 并自动调度计算节点来处理相应的数据块. 作业和任务调度功能主要负责分配和调度计算节点 (Map 节点或 Reduce 节点), 同时负责监控这些节点的执行状态, 并负责 Map 节点执行的同步控制, 这样就可以实现协同过滤的并行化.

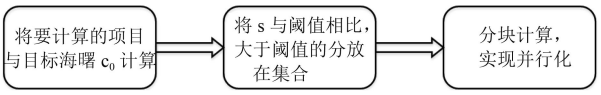

图 2 推荐过程流程图

3.2 基于 MapReduce 的协同过滤算法

本文研究的用户-类矩阵中的数据反映了用户对项目的喜爱程度, 但是其中的数据可能会有一定的差异, 所以基于 MapReduce 就在一定程度上可以解决这个问题. 并且可以计算用户相似度, 可以查找领居, 进行兴趣推荐, 那么就需要多轮 MapReduce 任务.

(1) 第 1 轮 Map 任务: 原始的用户数据处理后会统一存放在 HBase 数据库的信息表中, 这里面会有一系列的用户对项目的评分. Map 任务是将用户的评分信息读取出来 (String), 以<UserID, String>的形式读入数据, 经过对字符串分析, 挖掘客户信息, 以<UserID, Rating>的形式输出给接下来的 Reduce 任务.

(2) 第 1 轮 Reduce 任务: 此任务的输入就是上面 Map 的输出, 形式为<UserID, Rating…>, Reduce 读取任务后, 对序列求平均数, 作为新的输出, 以<UserID, Average Rating>的形式.

(3) 第 2 轮 Map 任务: 此任务的输入也是上一任务的输出, 分别以<UserID, String>和<UserID, Average Rating>方式输入, 经处理后, 以<UserID, ClassID+Rating>和<UserID, @Average Rating>的形式输出, 其中@符号是为了区分.

(4) 第 2 轮 Reduce 任务: 此任务的输入是<UserID, ClassID+Rating…, @ Average Rating >的形式输.

(5) 第 3 个 MapReduce 任务: 这个任务是对上一个 MapReduce 的结果做出整理, 计算用户相似度以及邻居, 并且进行排序, 然后得到我们想要的矩阵集合.

(6) 第 4 个 MapReduce 任务: 该任务是结合信息表中的信息和上一任务计算的用户相似度和邻居, 输出<(UserID + ClassID), (Rating + Similarity)>, 然后完成预测评分.

(7) 第 5 个 MapReduce: 对最终的数据进行排序, 通过找到预测评分最高的 k 个类得出推荐结果.

3.3 预测评分原理

在协同过滤最后一步就是产生推荐, 推荐的过程

中要对用户未评分的项目进行预测, 预测的方法如下, 我们可以用公式表示[7]:

$$prediction(u,i)=\frac{\sum_{n\in N} similarity(i,n)*rating(u,n)}{\sum_{n\in N}|(similarity(i,n))|} \quad (4)$$

其中 u 表示用户, i 表示未被用户评分过的项目, n 为与项目 i 相似的项目, N 是与项目 i 相似的项目集合.

上面公式中我们将进行一个矩阵的乘法, 在基于 MapReduce 框架中, 我们要先对矩阵分块处理, 使得不同模块在不同节点进行处理, 因此在利用预测评分公式前需要对输入的向量进行预处理, 会调用到 Mahout API 中的 partialMultiply 接口, 这个将会启动了两个 Map 任务和一个 Reduce 任务.

第一个 Map 负责读取训练集的用户向量. 主要是读取关键字为用户编号, 向量值为用户对各个项目的评分向量.

第二个 Map 任务是负责读取项目相似度向量, 关键字为项目索引, 值为相似度向量.

Reduce 任务的输入是上面两个 Map 任务的输出结果, 将关键字一致的向量聚集到一起, 组成一个新的矩阵, 然后计算最紧邻居集.

4 实验结果及分析

4.1 实验方案

本文实验数据采用开源的美国 Minnesota 大学 GronpLens 项目组提供的 Movielens(ml-lm) 数据集, 该数据集已经被广泛应用在推荐系统的测评中. 其包含了 6040 名用户对 3952 部电影的超过 1000 000 的评价, 分值为 1 到 5 的整数, 分数越高表示用户对电影的评价越高. 该数据集有 3 张数据表组成: Users(用户特征信息)、Movies(电影特征信息)、Ratings(评价信息). 每一条记录中包含用户标识、电影标识、评价数值和时间, 从中可以获取本文所需的用户基本信息、评价和时间等数据进行进一步的分析与处理.

本系统研究协同过滤算法的 MapReduce 并行化. 通过 Python 操作 HDFs 上的文件, 首先在 HDFs 上创建保存数据的文件夹; 然后 MapReduce 程序的每个输出文件作为下一个 MapReduce 的输入文件.

本实验中, 主要考察传统的基于用户的利用余弦相似度计算的协同过滤和在此基础上的协同过滤并行化的推荐质量. 实验分别为一台 PC 构建的单机模式以及三个节点构成的 Hadoop 集群. 本文主要通过两种方法来证实并行化的基于 Hadoop 的协同过滤算法的推荐系统及其推荐结果比传统的协同过滤算法的推荐系统其推荐结果更能符合用户的预期, 与此同时比较单节点和 3 个节点的运行时间以及加速比.

一种方法就是用运行时间作为衡量算法扩展性的指标, 同时也采用了加速比来衡量算法的扩展性, 加速比定义为单机运行时间和 Hadoop 集群的运行时间的比值, 计算公式如下:

$$speed=\frac{time_s}{time_h} \quad (5)$$

另外一种就是根据平均绝对误差协 MAE 值, MAE 的值越小, 表明算法的推荐精度越高. 这种方法就是衡量推荐与真实的用户赋给值的偏差. 对于每一对评分预测数据<p_i, q_i>的具体误差也就是$|p_i-q_i|$进行处理. 平均绝对误差的计算方法是先计算 N 对评分-预测数据对的误差之和, 然后计算平均值, 如下式:

$$MAE=\frac{\sum_{i=1}^{N}|p_i-q_i|}{N} \quad (6)$$

一般来说, 平均绝对误差越小, 推荐结果越准确, 系统性能就越好.

4.2 预测评分原理

(1) 一方面, 选取平均绝对误差协 (MAE) 作为算法推荐质量的评价指标, MAE 的值越小, 表明算法的推荐精度越高. 从 1 MB 数据集中分别选取数据集为 1000、10 000、15 000、20 000、40 000 和 60 000 作为 6 组实验数据, 分别用余弦相似度和修正余弦相似度以及本文方法 (利用两个节点并行) 进行 MAE 值比较, 如图 3 所示.

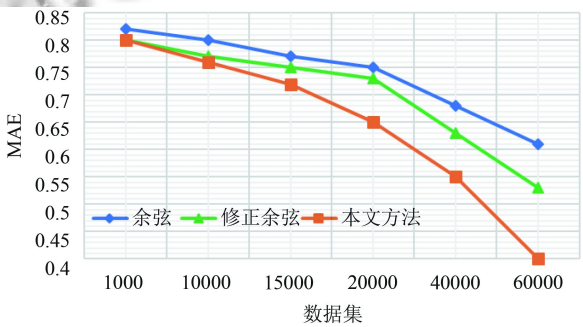

图 3 MAE 值

从图 3 可以看出, 实验数据集越稀疏, MAE 值越大, 这就说明数据越多, 可参考的就越多, 平均绝对误差也就越小; 当然, 我们将单机模式与并行模式作比较, 也不难发现在 Hadoop 环境下 MAE 的值也会相应的减少, 可以得知修正的余弦相似度是比传统的余弦相

似度这种算法更加提升了推荐质量, 并且基于 Hadoop 平台, 推荐性能也会随着节点数增加而得到改善.

(2) 接下来分析余弦相似性的协同过滤与并行化的协同过滤的运行时间, 因此我们将给出单节点、两个节点和三个节点的运行时间以及加速比的对比情况.

从图 4 可以看出, 随着数据集的增多, 运行时间也会随之增加, 相比单节点、两个节点以及三节点的 Hadoop 的运行情况, 总体来说, 在数据集很大的情况下并行能明显缩短时间, 这也足以说明基于 MapReduce 的协同过滤并行化推荐更适合于大数据的推荐系统. 但是我们发现在数据集很小的情况下, 并行化并没有显示出更加大的优势, 这是因为, 在数据集小的情况下, Hadoop 并行还需要进行资源配置, Map 和 Ruduce 之间进行数据传输也需要消耗时间, 因此在数据集很小的情况下就没必要进行并行.

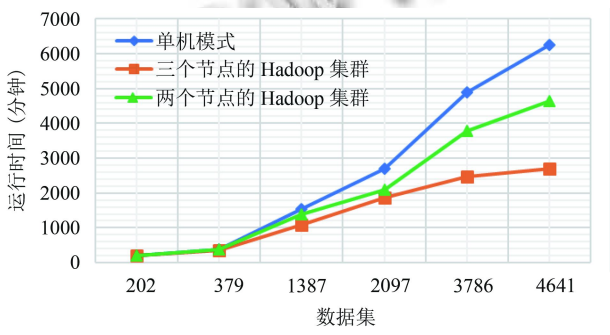

图 4 不同节点的运行时间

(3) 通过上述实验, 我们计算加速比进行比较如图 5 所示.

从图 5 我们可以看出, 并行化以后的协同过滤很大程度上的增大了运行时间的加速比, 并且当数据集越大的时候, 加速比就越大, 也就是说并行化对越大的数据集越有优化作用. 我们还看出, 理想的状况下, 要想充分地利用 Hadoop 集群的运算能力, Map 的个数最好是 Hadoop 的节点倍数, 这样还可以防止机器被闲置.

经过以上实验的分析, 我们可以得出基于 Hadoop 的协同过滤对较大的数据集能有成效的优化运算能力, 从而提升推荐速度. 随着现在互联网规模的不断扩大, 这一发现对于推荐系统具有极大的应用, 我们可以通过适当的增加 Hadoop 集群节点来提高推荐效率.

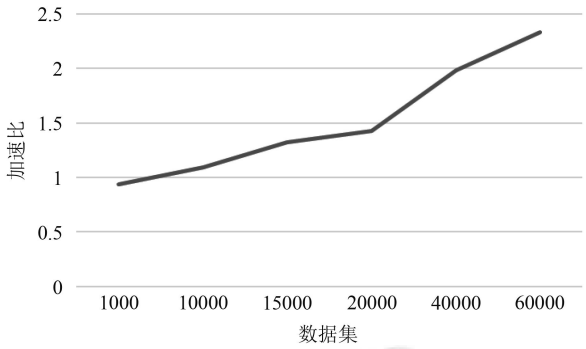

图 5 不同数据集的加速比

5 结论

本文介绍了传统基于项目和用户的协同过滤算法以及 Hadoop 平台的 MapReduce 编程模型, 分析了基于 Hadoop 平台的协同过滤算法. 实验验证在大规模数据的情况下, 采用基于 Hadoop 的 MapReduce 编程模型进行并行化的协同过滤算法可提高推荐效率和推荐精度.

参考文献

1 李涛, 王建东, 叶飞跃, 等. 一种基于用户聚类的协同过滤推荐算法. 系统工程与电子技术, 2007, 29(7): 1178–1182.

2 Xue GR, Lin CX, Yang Q, et a1. Scalable collaborative filtering using cluster-based smoothing. Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Salvador, Brazil. 2005. 114-121.

3 Zhao ZD, Shang MS. User-based collaborative-filtering recommendation algorithms on hadoop. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining. Phuket, Thailand. 2010. 478–481.

4 Pan R, Zhou YH, Cao B, *et al*. One-class collaborative filtering. Proceedings of the Eighth IEEE International Conference on Data Mining. Pisa, Italy. 2008. 502–511.

5 李灿. 基于 Hadoop 的并行化协同过滤推荐算法研究. [硕士学位论文]. 杨凌: 西北农林科技大学, 2016.

6 肖强, 朱庆华, 郑华, 等. Hadoop 环境下的分布式协同过滤算法设计与实现. 现代图书情报技术, 2013, (1): 83–89. [doi: 10.11925/infotech.1003-3513.2013.01.13]

7 李状. 基于 Hadoop 的协同过滤推荐算法的设计与实现[硕士学位论文]. 南京: 南京邮电大学, 2016.