文章编号: 1007-5321(2016)增-0037-05 DOI: 10.13190/j.jbupt.2016.s.009

基于 Spark 的 UCSLIM 推荐算法研究及实现

杨 娟, 张鹏业
(北京邮电大学 智能通信软件与多媒体北京市重点实验室, 北京 100876)

摘要: 稀疏线性(SLIM)推荐算法侧重于通过挖掘物品与物品之间的关系进而产生推荐结果. 为了提高推荐质量,借鉴了 SLIM 算法和协同过滤算法的思想,将用户划分为用户集合,进一步挖掘用户与用户集合之间的隐含关系,并综合考虑用户与用户相关性、用户与用户集合相关性这两个因素,提出了融合用户集合关系的稀疏线性(UCSLIM)推荐算法. 实验结果表明,UCSLIM 算法能够提高推荐结果质量. 同时为了提高算法的执行效率,分别在 Spark 和 Hadoop 云计算平台上实现了 UCSLIM 并行推荐算法,并通过实验表明,UCSLIM 的 Spark 版本具有更高的计算效率.

关 键 词: TopN 推荐; 稀疏线性; 用户集合; Spark
中图分类号: TP391.3 **文献标志码:** A

Design and Implementation of Parallel UCSLIM Algorithm Based on Spark

YANG Juan, ZHANG Peng-ye
(Key Laboratory of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: A top-N recommendation method—user class sparse linear methods (UCSLIM) based on sparse linear method (SLIM) was proposed. In order to improve the quality of recommendation, we learn from the idea of SLIM algorithm and collaborative filtering algorithm. The users are divided into different sets. So the correlation was analyzed between the user and the set of users and the correlation between user and user. Based on these two factors, UCSLIM was proposed. Experiments show that, compared with SLIM, UCSLIM can improve the quality of results. Furthermore, in order to improve the computational efficiency, the UCSLIM in Hadoop and Spark was implemented. Experiments show that the implementation by Spark has higher efficiency than that of Hadoop.

Key words: TopN recommender system; user class sparse linear methods; user class; Spark

目前,TopN 推荐系统已经得到广泛的应用与发展,稀疏线性(SLIM,sparse linear method)[1]推荐算法是一种基于模型的 TopN 推荐算法. 此算法先计算出物品之间的相关系数,然后将用户物品评分与物品相关系数相乘,从而预测出用户对其他物品的评分.

在借鉴 SLIM 算法思想的基础上,提出一种新的推荐算法——UCSLIM 算法. UCSLIM 算法综合考虑了用户-用户之间以及用户-用户集合之间的相关关系. 同时为了解决单机程序运行速度慢的问

收稿日期: 2015-09-17
基金项目: 国家高技术研究发展计划(国家 863 计划)项目(2015AA050204)
作者简介: 杨 娟(1972—),女,博士,副教授, E-mail: yangjuan@bupt.edu.cn.

题,同时在 Spark[2] 和 Hadoop 平台上实现了 UCSLIM 推荐算法的并行化版本,最后给出实验结果以及结论.

1 UCSLIM 算法

1.1 符号定义

符号 u 和 t 分别代表用户和物品,u_i 和 t_j 分别代表用户 i 和物品 j. 用户全集和物品全集分别用 U 和 T 表示,用户数量为 $n(|U| = n)$,物品数量为 $m(|T| = m)$. 所有矩阵均用大写字母表示,如 $\boldsymbol{A}$ 表示用户-物品评分矩阵,a_{ij} 表示用户 i 对物品 j 的评分. 列向量用加粗小写字母表示,如 $\boldsymbol{a}_j$,行向量用 $\boldsymbol{a}_i^{\mathrm{T}}$ 表示.

1.2 SLIM 算法简介

SLIM 算法通过式(1)预测用户 i 对物品 j 的评分有

$$\bar{a}_{ij} = \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{w}_j \tag{1}$$

其中:$\boldsymbol{a}_i^{\mathrm{T}}$ 为用户-物品评分矩阵 $\boldsymbol{A}$ 的第 i 行,即用户 i 对所有物品的评分记录;$\boldsymbol{w}_j$ 为物品 j 和其余物品之间的相关系数. 从式(1)可以看出,此算法是通过分析物品与物品之间的相关系数进而预测用户对物品的评分.

1.3 UCSLIM 算法

1.3.1 算法思想

在借鉴 SLIM 算法和协同过滤算法[3-4] 思想的基础上,提出了一种综合考虑个体(用户与用户)维度间的相关性以及个体和集合(用户与用户集合)之间的隐含相关性的推荐算法——UCSLIM 算法.

UCSLIM 算法首先会将用户划分为用户集合,然后计算出用户之间的相关系数以及用户与用户集合的相关系数. 在预测用户 i 对物品 j 的评分时,会同时将用户 i 和其他用户的相关性以及用户 i 和其他用户集合的相关性都考虑在内.

1.3.2 划分用户集合

在划分用户集合时,会将最相似的 C 个用户划分到同一个集合中. C 的取值不宜过大,C 的值越大,产生的用户集合数量越少,反而会影响对用户集合有效信息的挖掘,故采取 $C = 2$. 采用余弦相似度来衡量用户之间的相似度,在计算出所有用户之间的相似度后,对于用户 i,选取其最相似的用户 q,将用户 i 和用户 q 划分到同一个用户集 k 中. 以此类推,直到所有用户都找到了属于自己的用户集合,从而形成了一个用户集-物品矩阵 $\boldsymbol{A}'$. 在 $\boldsymbol{A}'$ 中,如果用户集合 k 中的用户 i 或用户 q 对物品 j 评分过,则 $a'_{kj} = 1$,否则为 0. 经过以上计算即可得出完整的用户集-物品评分矩阵 $\boldsymbol{A}'$. 因为采取的是每 2 个用户划分为一个用户集合,故矩阵 $\boldsymbol{A}'$ 的大小为 $(n/C) \times m$.

1.3.3 UCSLIM 计算模型

UCSLIM 算法通过式(2)预测用户 i 对物品 j 的评分有

$$\bar{a}_{ij} = \boldsymbol{r}_i^{\mathrm{T}} \boldsymbol{a}_j + \boldsymbol{r}_i'^{\mathrm{T}} \boldsymbol{a}'_j \tag{2}$$

其中:$\boldsymbol{r}_i^{\mathrm{T}}$ 为一个维度为 n 的行向量,表示用户 i 和其他用户的相关系数;$\boldsymbol{a}_j$ 为用户评分矩阵 $\boldsymbol{A}$ 的第 j 列,即所有用户对于物品 j 的评分;$\boldsymbol{r}_i'^{\mathrm{T}}$ 为一个维度为 n/C 的行向量,表示用户 i 和其他用户集的相关系数;$\boldsymbol{a}'_j$ 为用户集-物品评分矩阵 $\boldsymbol{A}'$ 的第 j 列,即所有用户集合对于物品 j 的评分. 将式(2)扩展到矩阵相乘的形式为

$$\bar{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{A} + \boldsymbol{R}'\boldsymbol{A}' \tag{3}$$

其中:$\boldsymbol{R}$ 为大小为 $n \times n$ 的用户-用户相关系数矩阵,$\boldsymbol{R}'$ 为大小为 $n \times (n/C)$ 的用户-用户集相关系数矩阵,$\boldsymbol{A}'$ 是大小为 $(n/C) \times m$ 的用户集-物品评分矩阵,即根据原始的用户-物品矩阵计算出的新的用户集-物品矩阵. 通过如下公式可以计算出 $\boldsymbol{R}$ 和 $\boldsymbol{R}'$.

$$\min \frac{1}{2} \| \boldsymbol{A} - (\boldsymbol{R}\boldsymbol{A} + \boldsymbol{R}'\boldsymbol{A}') \|_{\mathrm{F}}^2 + \frac{\beta_1}{2} \| \boldsymbol{R} \|_{\mathrm{F}}^2 + \theta_1 \| \boldsymbol{R} \|_1 + \frac{\beta_2}{2} \| \boldsymbol{R}' \|_{\mathrm{F}}^2 + \theta_2 \| \boldsymbol{R}' \|_1$$

$$\text{约束} \quad \boldsymbol{R} > 0 \text{ and } \mathrm{Diag}(\boldsymbol{R}) = 0, \quad \boldsymbol{R}' > 0 \text{ and } \mathrm{Diag}(\boldsymbol{R}') = 0 \tag{4}$$

其中:$\frac{1}{2} \| \boldsymbol{A} - (\boldsymbol{R}\boldsymbol{A} + \boldsymbol{R}'\boldsymbol{A}') \|_{\mathrm{F}}^2$ 评估了预测值($\boldsymbol{R}\boldsymbol{A} + \boldsymbol{R}'\boldsymbol{A}'$)和实际值($\boldsymbol{A}$)之间的偏离程度;$\| \boldsymbol{R} \|_1 = \sum_{i=1}^{m} \sum_{j=1}^{m} |r_{ij}|$ 是 $\boldsymbol{R}$ 的 L1 范式,即 L1 正则化约束,$\| \boldsymbol{R} \|_{\mathrm{F}}$ 为矩阵 $\boldsymbol{R}$ 的 F 范式,即 L2 正则化约束. L1 和 L2 正则化约束的作用是防止计算得到的模型过拟合[5]. 常量 $\beta_{1,2}$ 和 $\theta_{1,2}$ 分别是 L1 正则化和 L2 正则化约束参数,一般来说,当常量 $\beta_{1,2}$ 和 $\theta_{1,2}$ 越大,其正则化程度越高;约束条件 $\boldsymbol{R} > 0$ 和 $\boldsymbol{R}' > 0$ 保证了用户集合和用户集合以及用户和用户之间的正相关性,$\mathrm{Diag}(\boldsymbol{R}) = 0$ 和 $\mathrm{Diag}(\boldsymbol{R}') = 0$ 保证了其对角线元素都为 0,即保证用户或用户集合对自身的推荐程度为 0.

求解式(4)本质上是一个最优化问题的求解,采用随机梯度下降法[6]求解. 在求解梯度过程中,向量的 L1 正则化约束在 0 附近不可导,采用 Tsuruoka 等[7]提出的方法处理 0 附近不可导问题.

1.3.4　并行求解 UCSLIM 算法

由线性代数知识可知,当使用式(4)求解矩阵 R 和矩阵 R' 时,矩阵 R 和矩阵 R' 内的每一行之间不存在直接相关性,故可以将式(4)转化为一组规模更小的相同的求解问题,其优化之后为式(5).

$$\min \frac{1}{2}\| \boldsymbol{a}_i^{\mathrm{T}} - (\boldsymbol{r}_i^{\mathrm{T}}\boldsymbol{A} + \boldsymbol{r}_i'^{\mathrm{T}}\boldsymbol{A}') \|_{\mathrm{F}}^2 + \frac{\beta_1}{2}\| \boldsymbol{r}_i^{\mathrm{T}} \|_{\mathrm{F}}^2 + \theta_1 \| \boldsymbol{r}_i^{\mathrm{T}} \|_1 + \frac{\beta_2}{2}\| \boldsymbol{r}_i'^{\mathrm{T}} \|_{\mathrm{F}}^2 + \theta_2 \| \boldsymbol{r}_i'^{\mathrm{T}} \|_1$$

$$\text{约束} \quad \boldsymbol{r}_i^{\mathrm{T}} > 0 \text{ and } \boldsymbol{r}_{ii}^{\mathrm{T}} = 0 \qquad \boldsymbol{r}_i'^{\mathrm{T}} > 0 \text{ and } \boldsymbol{r}_{ii}'^{\mathrm{T}} = 0 \tag{5}$$

其中: $\boldsymbol{a}_i^{\mathrm{T}}$ 为矩阵 $\boldsymbol{A}$ 的第 i 行,$\boldsymbol{r}_i^{\mathrm{T}}$ 为矩阵 $\boldsymbol{R}$ 的第 i 行,$\boldsymbol{r}_i'^{\mathrm{T}}$ 为矩阵 $\boldsymbol{R}'$ 的第 i 行. 通过此种变换,只需要并行求解矩阵 $\boldsymbol{R}$ 和矩阵 $\boldsymbol{R}'$ 的每一行,然后将所有行的结果合并成一个完整矩阵即可.

1.3.5　算法复杂度

对于 UCSLIM 算法,在划分用户集合时,要计算所有用户之间的相似度,时间复杂度为 $O(n^2)$. 在计算式(5)时,假设进行了 k 次迭代,每次迭代计算目标函数值时,复杂度最高的部分为将维度为 n 的 $\boldsymbol{r}_i^{\mathrm{T}}$ 与矩阵 $\boldsymbol{A}(n \times m)$ 相乘,其复杂度为 $O(nm)$. 综上,UCSLIM 算法总的时间复杂度为 $O(knm + n^2)$. 空间上主要是矩阵 $\boldsymbol{R}(m \times m)$ 和矩阵 $\boldsymbol{A}(n \times m)$ 占用空间,故其空间复杂度为 $O(nm + m^2)$.

2　实验与分析

2.1　实验平台介绍

实验环境是由 5 台配置相同的服务器组成. Hadoop 版本为 2.60,Spark 版本为 1.3.0. 每台服务器的配置情况为: 双 CPU,24 核,内存 64 GB; 每个核的型号为: Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz.

2.2　数据集以及预处理

为了验证算法的有效性,采用了 5 个数据集: movieLens100K、movieLens1M 电影评分数据集,lastFM 音乐收听数据集,news 新闻阅读数据集,BookCrossing(BX)图书阅读数据集. 5 个数据集的详细信息如表 1 所示.

表 1　数据集详细信息

数据集	用户数	物品数	评分项	稠密度/%
mL100K	943	1 682	55 375	3.49
lastFM	1 865	2 000	55 963	1.50
news	1 149	5 035	41 057	0.71
ML1M	6 040	3 952	71 6307	3.00
BX	5 698	6 305	86 237	0.24

2.3　评价标准

2.3.1　推荐质量评价标准

实验采用 TopN 准确率作为评价指标. 准确率是通过计算用户在 TopN 列表中实际打分的物品数量的比例来衡量推荐系统的预测质量. 准确率越高,意味着其推荐质量越高. 假设算法对于用户 X 推荐的 N 个物品为 $T\{t_1, t_2, \cdots, t_N\}$,对应的用户 X 的实际购买物品为 $F\{f_1, f_2, \cdots, f_N\}$,则该推荐算法对于用户 X 的准确率为

$$P = \frac{|T \cap F|}{|T|} \tag{6}$$

2.3.2　运行效率评价标准

采用加速比来评价 UCSLIM 算法在不同的云计算平台(Spark 和 Hadoop)上的计算效率. 加速比是指同一个任务在单机系统和并行系统中运行消耗的时间的比率,一般用来衡量系统并行化的性能和效果. 其计算公式为

$$S_{\mathrm{p}} = \frac{U_1}{U_n} \tag{7}$$

其中: U_1 为任务在单机系统中的运行时间,U_n 为任务在并行系统中的运行时间. S_{p} 的值越大,表示其加速性能越好,即并行效率越高.

2.4　实验结果及分析

2.4.1　3 种算法在不同 TopN 时的推荐效果

为了验证算法的有效性,采用了 SLIM、itemKNN 两种算法作为 UCSLIM 算法的对比算法,每种算法设定的参数取值均为多次实验后确定的最优值. 为了排除实验结果的偶然性,实验采用 5 折交叉验证的方式,其实验结果如图 1 ~ 图 5 所示.

从实验结果可以看出,UCSLIM 算法在各个数据集上的推荐效果都要优于 SLIM 算法. 3 种算法在 news 数据集上表现最好,因为相对于电影、图书等领域,用户偏好的新闻分类较为固定,故推荐算法能够较为准确地分析出用户喜好. 在其余的数据集中,mL100K 数据集的稠密度最高. 稠密度越高,说

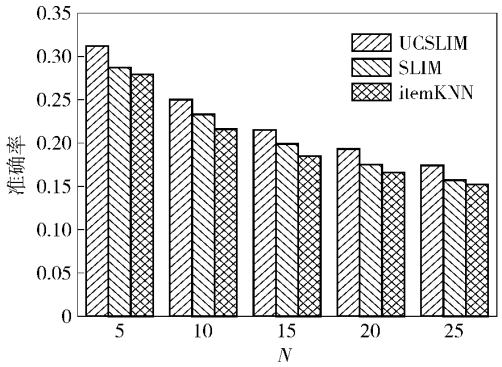

图1 在mL100K数据集中各推荐算法效果

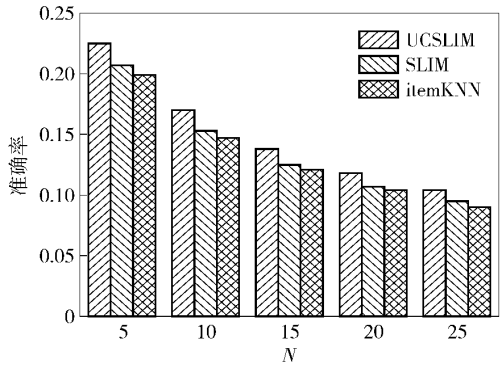

图2 在lastFM数据集中各推荐算法效果

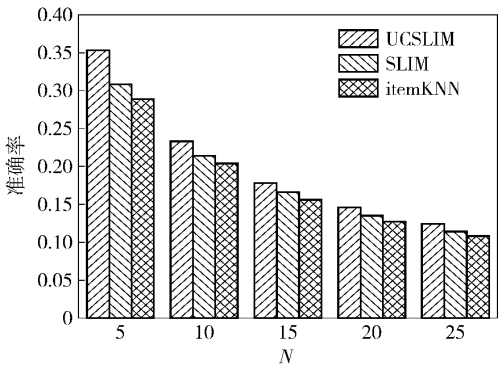

图3 在news数据集中各推荐算法效果

明其中包含有效的信息越多,故其推荐效果要优于其他几个数据集. 反之,BX数据集的稠密度最低,其推荐效果是相对较差的.

2.4.2 运行效率比较

分别在Spark和Hadoop两种云计算平台上实现了UCSLIM算法,其在两个平台上的运行时间以及加速比如表2所示.

从表2可以看出,Spark的加速效果要优于Hadoop. 一方面是因为Hadoop的map阶段需要将文件写入到硬盘中,并且会按照key排序,但是排序操作在计算用户相关系数矩阵时是没有必要的. Spark则是将数据保存在内存中,且没有排序操作,从而节省了时间. 另一方面则是并行程度不同,Hadoop是进程级并行,Spark则是线程级并行,并行程度更高,故在计算用户相关系数矩阵时,能够更高效地并行计算.

图4 在mL1M数据集中各推荐算法效果

图5 在BX数据集中各推荐算法效果

表2 运行时间以及加速比

时间/s	mL100K	lastFM	news	mL1M	BX
Spark	4	9	6	87	75
Hadoop	37	65	28	300	192
单机	50	3 415	905	41 009	12 037
Sp(Spark)	12.5	397.4	150.8	471.4	160.5
Sp(Hadoop)	9.25	52.5	32.3	136.7	62.7

3 结束语

在借鉴SLIM推荐算法思想的基础上,提出了基于用户集合的UCSLIM算法. UCSLIM算法将用户划分为用户集合,并将用户-用户、用户-用户集合这两个隐含关系加入到算法当中,更加充分地挖掘了原始数据中的隐含信息,并通过实验证明UCS-

LIM 算法比 SLIM 算法推荐质量更高. 同时，在 Spark 和 Hadoop 两个云计算平台上实现了 UCSLIM 算法，并通过实验表明 UCSLIM 算法的 Spark 版具有更高的计算效率. 提出的 UCSLIM 算法也存在可改进之处，如在划分用户集合时，可使用标准的频繁项集划分方式，这样会增加预处理时的时间复杂度，但是相应的集合划分可能更为精确.

参考文献：

[1] Ning X，Karypis G. SLIM：sparse linear methods for Top-N recommender systems [C] //2011 11th IEEE International Conference on Data Mining. Vancouver，Canada：IEEE ICDM，2011：497-506.

[2] Zaharia M，Chowdhury M，Das T，et al. Fast and interactive analytics over Hadoop data with Spark [J]. USENIX，2012，37(4)：45-51.

[3] Marlin B M. Modeling user rating profiles for collaborative filtering [C] // Advances in Neural Information Processing Systems 16. Vancouver，Canada：NIPS，2003：627-634.

[4] Sarwar B，Karypis G，Konstan J，et al. Item-based collaborative filtering recommendation algorithms [C] // Proceedings of the 10th International Conference on World Wide Web. Hong Kong：ACM，2001：285-295.

[5] Zou H，Hastie T. Regularization and variable selection via the elastic net [J]. Journal of the Royal Statistical Society：Series B (Statistical Methodology)，2005，67(2)：301-320.

[6] Zhang T. Solving large scale linear prediction problems using stochastic gradient descent algorithms [C] // Proceedings of the Twenty-first International Conference on Machine Learning. New York，USA：ACM，2004：116.

[7] Tsuruoka Y，Tsujii J，Ananiadou S. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty [C] // Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP：Volume 1-Volume 1. Suntec，Singapore：Association for Computational Linguistics，2009：477-485.