DOI：10.11830/ISSN.1000-5013.201703071

Spark 平台下 KNN-ALS 模型推荐算法

邹小波，王佳斌，詹敏

（华侨大学 工学院，福建 泉州 362021）

摘要： 考虑 Spark 大数据平台内存计算框架在迭代计算的优势，提出 Spark 平台下 KNN-ALS 模型的推荐算法. 针对矩阵分解算法只考虑隐含信息而忽视相似度信息的缺陷，将相似度信息加入评分预测中，并采用适合并行化的交替最小二乘法进行模型最优. 在 MovieLens 数据集上的实验表明：该算法能够提高协同过滤推荐算法在大数据集下的处理效率，且加速比也达到并行处理的线性要求，相比其他方法有较好的精度.

关键词： 推荐算法；KNN-ALS 模型；协同过滤；Spark 平台；矩阵分解

中图分类号： TP 391.1 **文献标志码：** A **文章编号：** 1000-5013(2019)02-0264-05

Recommendation Algorithm of KNN-ALS Model Based on Spark Platform

ZOU Xiaobo, WANG Jiabin, ZHAN Min

(Engineering Institude, Huaqiao University, Quanzhou 362021, China)

Abstract: Taking into account the memory computing advantage of Spark framework in iterative computation, the KNN-ALS model of recommendation algorithm based on Spark is proposed in this paper. The matrix factorization algorithm only considers the implicit information but ignores the similarity information, the model adds the similarity information into the rating prediction and then use the method of alternating least squares to optimize the model. From the experiments on the MovieLens dataset, the algorithm can improve the processing efficiency of the collaborative filtering algorithm in large data set, and also got a regular parameter of speedup in parallel processing. Furthermore, the proposed model have better accuracy than other methods.

Keywords: recommendation algorithm; KNN-ALS model; collaborative filtering; Spark platform; matrix factorization

随着大数据时代的发展，人们从诸多纷杂的数据中找到自己想要内容的需求越来越强烈，推荐系统已经成为工业界的研究热点[1]. 推荐算法从提出到发展至今主要分为 3 类：基于内容的推荐、协同过滤推荐、基于知识的推荐等[2-5]. 推荐领域的推荐算法研究主要集中在基于模型的协同过滤推荐上[6-12]，且使用最多的是矩阵分解技术[13-15]. 在诸多的 Netflix 推荐算法[16]竞赛中，基于矩阵分解模型的协同过滤推荐相比其他的推荐算法能产生更加准确的推荐. 在大数据下，单机模式进行算法的计算已渐渐不能满足实时推荐的需求，因此，分布式推荐算法的研究成为一个新的方向. 在已有的分布式研究中，大多是基于 Hadoop[17]平台进行推荐算法的并行化设计，但是 MapReduce[18]框架在迭代计算时效率低下，影响算法执行速度. 大数据平台 Spark[19]是 UC Berkeley AMP Lab 开源的通用分布式内存计算框架，通过

收稿日期： 2017-03-28

通信作者： 王佳斌(1974-)，男，副教授，博士，主要从事物联网、云计算、大数据、智能仪器的研究. E-mail：fatwang@hqu.edu.cn.

基金项目： 国家自然科学基金青年科学基金资助项目(61505059)；福建省厦门市科技局产学研科技创新项目(3502Z20173046)；华侨大学研究生科研创新能力培养计划项目(1511422010)

将数据缓存至内存进行计算，极大提高了数据的读写速率，并在多种情景下都能进行数据处理. 本文考虑在大数据内存计算 Spark 平台下，研究基于矩阵分解模型的推荐算法在 Spark 平台上的移植，将相似度的推荐算法[18]与矩阵分解模型的推荐算法结合，以解决大数据背景下数据庞杂及矩阵分解算法在单机下执行效率低的问题.

1　基于矩阵分解技术的 KNN-ALS 模型

1.1　矩阵分解

矩阵分解的核心是将用户-物品评分矩阵从高维分解为若干低维矩阵的乘积组成. 区别于基于内存的推荐算法，该类算法是一种基于模型的评分预测算法，其实际意义在于将评分矩阵中缺失的评分通过回归的方式填充.

假设具有 N 个用户 M 个项目，先把矩阵的缺失值填充成 $\boldsymbol{R}'$；然后，将该矩阵分解，即将 $\boldsymbol{R}'$ 分解为用户矩阵 $\boldsymbol{P}\in\mathbf{R}^{N\times K}$ 和物品矩阵 $\boldsymbol{Q}\in\mathbf{R}^{N\times K}$，$K$ 为用户与项目间的主题.

通过分解后的矩阵估计评分为

$$\hat{r}_{u,i}=\boldsymbol{p}_u\boldsymbol{q}_i. \tag{1}$$

风险函数构造为

$$C(\boldsymbol{p},\boldsymbol{q})=\min\sum_{(u,i)\in k}^{n}(\hat{r}_{u,i}\boldsymbol{p}_u\boldsymbol{q}_i^{\mathrm{T}})^2+\lambda(\|\boldsymbol{p}_u\|^2+\|\boldsymbol{q}_i\|^2). \tag{2}$$

式(2)中：λ 为正则化模型参数，用来避免过拟合问题；k 表示评分项，评分预测的目的就是将上述的损失函数最小化.

最优化中，最小化损失函数的方法一般有随机梯度下降法(SGD)[20]和最小二乘法(ALS)[21]. 从实现上看，梯度下降法更为简单且迭代的收敛速度更快，但由于其本身是序列化的，因而不容易实现并行化；而最小二乘法更容易实现并行化，这也是基于 Spark 平台利用 ALS 优化方法进行改进的依据.

1.2　基于 KNN-ALS 的推荐模型

Spark MLlib 中已有的对于协同过滤推荐算法的应用主要有基于矩阵分解的 ALS 算法. Spark ALS 算法的核心就是将稀疏矩阵进行分解为用户矩阵和物品矩阵，然后，交替使用最小二乘法进行用户与物品的特征向量更新使其误差平方和最小. KNN-ALS 算法针对传统的矩阵分解模型推荐算法忽略相似度信息的缺点，将 KNN 算法得到的相似度信息融入进 ALS 算法中的损失函数.

实际上，通过分析可以知道，有时评分系统与用户物品无关，而用户有些属性与物品无关，物品也有些属性和用户无关. 因此，在预测评分中加入偏置项，得到的预测公式为

$$\hat{r}_{u,i}=\mu+b_u+b_i+\boldsymbol{p}_u\boldsymbol{q}_i. \tag{3}$$

式(3)中：μ 为训练集中所有记录评分的全局平均数；b_u 是用户偏置项；b_i 是物品偏置项. 在对物品进行评分时，有的用户可能比较苛刻，评分普遍较低，有时某项物品本身质量高，与用户评价没有很大的关联，上述的偏置项是可以在 Spark 平台下进行并行化的统计所得. 因此，在评分的预测项中加入偏置项，能很好地提升算法的表现并利用并行化的优势.

根据式(2)中 ALS 模型训练的损失函数，在使用 ALS 模型求解用户矩阵和物品矩阵的过程中，发现并没有将用户或物品相似度考虑进去. 由此，将用户或物品相似度融入损失函数中，以减小系统误差，得到 KNN-ALS 推荐模型，相似度误差计算式为

$$U=\sum_{u\in N_u}(\boldsymbol{p}_{u,k}-\frac{\sum_{\boldsymbol{p}_v\in\mathrm{KNN}(\boldsymbol{p}_u)}(S_{\boldsymbol{p}_u\boldsymbol{p}_v}\boldsymbol{p}_{v,k})}{\sum_{\boldsymbol{p}_v\in\mathrm{KNN}(\boldsymbol{p}_u)}S_{\boldsymbol{p}_u\boldsymbol{p}_v}},\qquad V=\sum_{i\in N_i}(\boldsymbol{q}_{i,k}-\frac{\sum_{\boldsymbol{q}_j\in\mathrm{KNN}(\boldsymbol{q}_i)}(S_{\boldsymbol{q}_i\boldsymbol{q}_j}\boldsymbol{q}_{j,k})}{\sum_{\boldsymbol{q}_j\in\mathrm{KNN}(\boldsymbol{q}_i)}S_{\boldsymbol{q}_i\boldsymbol{q}_j}}. \tag{4}$$

式(4)中：N 表示用户和物品的集合；KNN($\boldsymbol{p}$)表示某个用户或物品的相似近邻；S 表示两者间的相似度；k 表示用户与物品的任意属性.

上述公式从相似用户或物品的角度进行用户-物品评分误差的计算，并将得到的相似度误差信息融入基于 ALS 的评分误差预测中，从而得到 KNN-ALS 算法. KNN-ALS 模型的损失函数计算式为

$$C(\boldsymbol{p},\boldsymbol{q})=\min\sum_{(u,i)\in k}^{n}(\hat{r}_{u,i}-\boldsymbol{p}_u\boldsymbol{q}_i^{\mathrm{T}})^2+\lambda(\|\boldsymbol{p}_u\|^2+\|\boldsymbol{q}_i\|^2)+U+V. \tag{5}$$

增加相似度信息的算法在Spark上的并行化优势在于:每个用户和物品的相似度可以通过并行计算得到用户物品相似度矩阵,构造ALS模型时,只需将相似度信息加入到损失函数中进行迭代计算.根据上述算法的计算步骤,可以得到整体算法的计算流程,如图1所示.

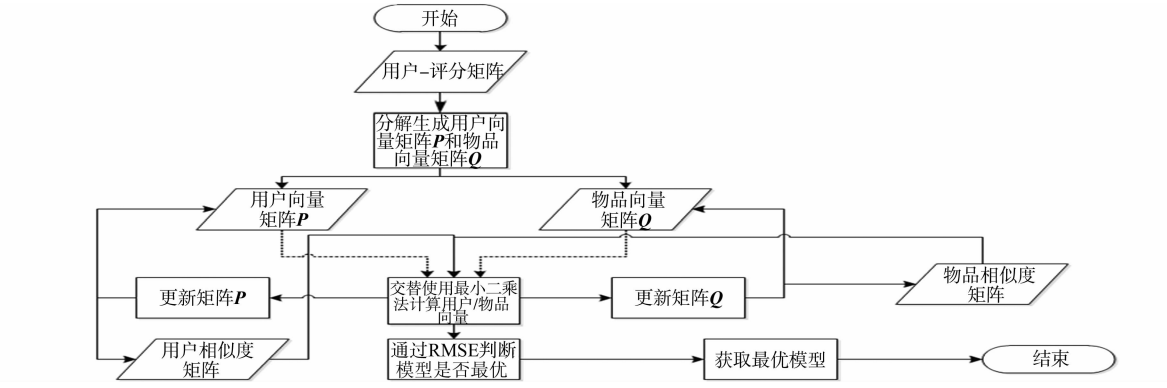

图1 并行化推荐算法的流程图

Fig. 1 Parallelization of proposed algorithm flow chart

2 实验设计与结果分析

实验使用3台主频3.3 GHz、内存为8 GB的主机组成的Spark集群,Spark部署在Ubuntu14.04系统下,并且使用最新的Spark2.0.2.实验数据集来自GroupLens的MovieLens数据集.在实验中使用的是标准数据集在分布式集群下进行推荐算法模型的训练,从运行时间、评分预测的均方根误差,以及并行化性能加速比进行分析.同时,为了测试文中所提算法在大数据集下的效能,使用Netflix的公开电影数据集进行实验验证.表1为实验中使用的数据集汇总.

表1 实验数据集

Tab. 1 Experimental data set

数据集	用户数	物品数	评分数	容量/MB
ml-100K	943	1 682	1.0×10^5	2
ml-1M	6 040	3 900	1.0×10^6	20
ml-10M	71 567	10 681	1.0×10^7	250
ml-20M	138 493	27 278	2.0×10^7	500
Netflix	480 189	17 770	1.0×10^8	3 072

2.1 运行时间

根据不同数据集在Spark集群下推荐模型的训练时间,得到集群和大数据集的运行时间(t),如图2,3所示.由图2可知:同一数据集的大数据平台Spark下,增加集群中主机数可以明显减少处理时间,从而提高算法的执行速度,而且集群主机数越多效果越明显.由图3可知:算法的耗时随着迭代次数的增加大致是呈线性增加的.

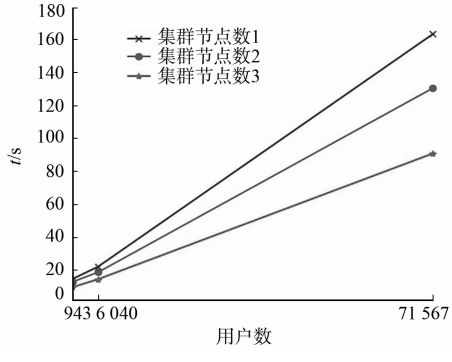

图2 集群运行时间

Fig. 2 Running times on clusters

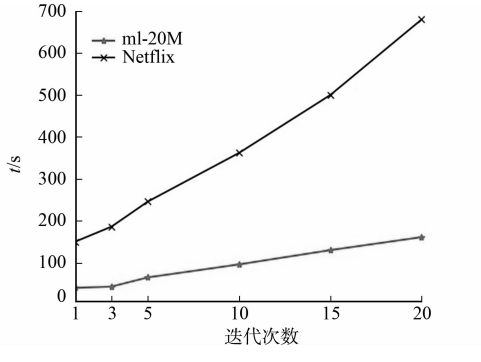

图3 大数据集运行时间

Fig. 3 Running times on big datasets

2.2 加速比

加速比通过单节点与多节点所耗时间之比计算算法的并行化性能好坏,其计算式为

$$v(p) = t_1/t_p, \qquad p = 1,2,3,\cdots. \tag{6}$$

式(6)中：t_1 为算法在单个节点上的运行所耗时间；t_p 为多个节点运行消耗的时间.

通过加速比表现并行化算法的性能，结果如图 4 所示.

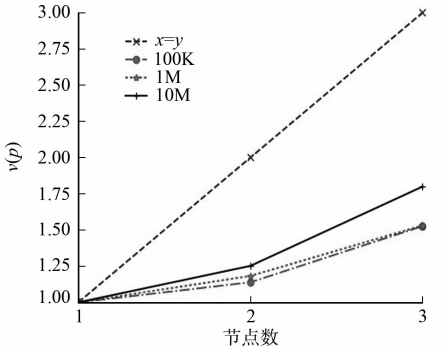

图 4　并行化推荐算法的加速比

Fig. 4　Parallelization recommended algorithm speedup

2.3　均方根误差

利用均方根误差(E_{RMS})作为评价参数，计算实际用户评分与预测用户评分之间的误差来衡量准确性，其计算式为

$$E_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(\boldsymbol{p}_i - \boldsymbol{r}_i)^2}. \tag{7}$$

式(7)中：N 为品评分数；$\boldsymbol{p}_i$ 为预测评分；$\boldsymbol{r}_i$ 为实际评分. E_{RMS} 值越小，代表误差越小，预测准确性越高.

上述并行化推荐算法在不同的参数下 E_{RMS} 的变化情况(Ranks=10)，如表 2 所示. 表 2 中：λ 为正则化系数. 由表 2 可知：不同参数训练的模型对 E_{RMS} 具有一定影响，在属性个数和正则化系数一定的情况下，迭代次数增多，E_{RMS} 会越来越小并趋于平稳，之后，正则化系数对 E_{RMS} 的影响会成为主要因素. 由大数据集上的变化情况也可以看出，迭代次数增大能够减小 E_{RMS}，迭代次数很少时，表现不如小数据集；而从数据集属性可知，这是由数据的稀疏性引起的，当增大到 20 次时，表现趋于稳定. 因此，所提算法在大数据与小数据上是有类似表现的，证明所提算法具有一定鲁棒性.

表 2　不同参数下并行化推荐算法的 E_{RMS} 变化

Tab. 2　E_{RMS} changes of parallelized recommendation algorithms with different parameters

数据集	迭代次数	λ		
		0.001	0.01	0.1
ml-100K	1	1.981 323	2.180 021	2.023 059
	5	0.730 234	0.712 914	0.792 341
	10	0.700 484	0.693 018	0.772 354
	20	0.695 818	0.675 462	0.682 456
	40	0.689 425	0.673 256	0.694 268
Netflix	1	2.800 325	3.133 657	3.701 477
	5	0.908 376	0.917 227	0.975 160
	10	0.798 455	0.805 930	0.848 851
	20	0.694 444	0.698 339	0.747 946

2.4　算法对比

为了比较改进后算法与其他算法的运行效果，在单机环境下，对同一数据集 ml-100K 进行算法测试. 选用的算法分别是 Spark ML 库中的 ALS 模型及利用 KNN 算法(选用 Pearson 相似度)进行基于近邻用户的推荐，评价指标仍然采用 E_{RMS}，结果如图 5 所示. 图 5 中：KNN 算法横坐标表示 K 值，2 个 ALS 模型表示最小化损失函数过程中的迭代次数，其他 2 个参数 λ 和 Ranks 分别使用了 0.1 和 10.

由图 5 可知：ALS 模型的精度较基于 KNN 算法的高，而加入了相似度信息后的 ALS 也减小了其预测的误差；在 E_{RMS} 指数下降的过程中，KNN-ALS 的收敛速度明显慢于 ALS，造成 KNN-ALS 算法耗时更多，这也是精度提升的代价.

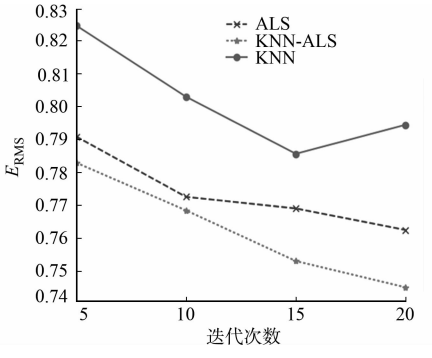

图 5　改进前后的算法对比

Fig. 5　Comparison of algorithms before and after improvement

3　结论

提出结合大数据处理与推荐技术进行推荐算法改进. 改进后的 KNN-ALS 算法在原有的 ALS 算法基础上加入 KNN 相似度信息. 实验表明：在 Spark 平台下的 KNN-ALS 有更快的运行效率，通过加速比的比较可知，算法有较好的并行化性能. 通过 E_{RMS} 参数可知：迭代次数的增加及正则化系数的调整有益于减小实验的误差.

通过对大数据集上的鲁棒性测试也说明算法有良好的表现. 由于算法依赖用户数据，存在数据稀疏和冷启动问题，如何在复杂的数据环境中保持算法一定的伸缩性及实时性，是算法在后续研究中需要补充和完善的地方.

参考文献:

[1] AMATRIAIN X. Past, present, and future of recommender systems: An industry perspective[C]// International Conference on Intelligent User Interfaces. Boston: ACM, 2016: 211-214. DOI: 10.1145/2959100.2959144.

[2] JANNACH D, FELFERNIDG A, FRIEDRICH G. 推荐系统[M]. 蒋凡, 译. 北京: 人民邮电出版社, 2013: 1-8.

[3] 孟祥武, 刘树栋, 张玉洁, 等. 社会化推荐系统研究[J]. 软件学报, 2015, 26(6): 1356-1372. DOI: 10.13328/j.cnki.jos.004831.

[4] KOREN Y, BELL R. Advances in collaborative filtering[M]. Berlin: Springer, 2011: 145-186. DOI: 10.1007/978-1-4899-7637-6_3.

[5] SU Xiaoyuan, KHOSHGOFTAAR T M. A survey of collaborative filtering techniques[J]. Advances in Artificial Intelligence, 2009(12): 1-19. DOI: 10.1155/2009/421425.

[6] SARWAR B, KARYPIS G, KONSTAN J, *et al*. Item-based collaborative filtering recommendation algorithms[C]// International Conference on World Wide Web. New York: ACM, 2001: 285-295. DOI: 10.1145/371920.372071.

[7] BILLSUS D, PAZZANI M J. Learning collaborative information filters[C]// Proceedings of the Fifteenth International Conference on Machine Learning. San Francisco: Morgan Kaufmann Publishers Inc, 1998: 46-54.

[8] SALAKHUTDINOV R, MNIH A. Probabilistic matrix factorization[C]// Proceedings of the 20th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc, 2007: 1257-1264.

[9] SALAKHUTDINOV R, MNIH A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo [C]// International Conference on Machine Learning. New York: ACM, 2008: 880-887. DOI: 10.1145/1390156.1390267.

[10] WANG Chong, BLEI D M. Collaborative topic modeling for recommending scientific articles[C]// Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2011: 448-456. DOI: 10.1145/2020408.2020480.

[11] WANG Hao, WANG Naiyan, YEUNG D Y. Collaborative deep learning for recommender systems[C]// Proceeding KDD'15 Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2015: 1235-1244. DOI: 10.1145/2783258.2783273.

[12] ZHANG Zike, ZHOU Tao, ZHANG Yicheng. Tag-aware recommender systems: A state-of-the-art survey[J]. Journal of Computer Science and Technology, 2011, 26(5): 767-777. DOI: 10.1007/S11390-011-0176-1.

[13] KOREN Y, BELL R, VOLINSKY C. Matrix factorization techniques for recommender systems[J]. Computer, 2009, 42(8): 30-37. DOI: 10.1109/MC.2009.263.

[14] SYMEONIDIS P. Matrix and tensor decomposition in recommender systems[C]// ACM Conference on Recommender Systems. Boston: ACM, 2016: 429-430. DOI: 10.1145/2959100.2959195.

[15] 张川. 基于矩阵分解的协同过滤推荐算法研究[D]. 长春: 吉林大学, 2013: 29-50.

[16] PARASCHAKIS D. Recommender systems from an industrial and ethical perspective[C]// ACM Conference on Recommender Systems. New York: ACM, 2016: 463-466. DOI: 10.1145/2959100.2959101.

[17] WHITE T. Hadoop 权威指南[M]. 3版. 华东师范大学数据科学与工程学院, 译. 北京: 清华大学出版社, 2015.

[18] 杜江, 张铮, 张杰鑫, 等. MapReduce 并行编程模型研究综述[J]. 计算机科学, 2015, 42(增刊1): 2635-2642.

[19] KARAU H, KONWINSKI A, WENDELL P, *et al*. Spark 快速大数据分析[M]. 王道远, 译. 北京: 人民邮电出版社, 2015: 187-209.

[20] 李卫平, 杨杰. 基于随机梯度矩阵分解的社会网络推荐算法[J]. 计算机应用研究, 2014, 31(6): 1654-1656. DOI: 10.3969/j.issn.1001-3695.2014.06.011.

[21] ZHOU Yunhong, WILKINSON D, SCHREIBER R, *et al*. Large-scale parallel collaborative filtering for the netflix prize[C]// Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management. Berlin: Springer-Verlag, 2008: 337-348. DOI: 10.1007/978-3-540-68880-8_32.

(责任编辑: 黄晓楠 **英文审校:** 吴逢铁)