软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(11):2981−2993 [doi: 10.13328/j.cnki.jos.004904]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于评分矩阵局部低秩假设的成列协同排名算法*

刘海洋, 王志海, 黄 丹, 孙艳歌

(北京交通大学 计算机与信息技术学院,北京 100044)

通讯作者: 王志海, E-mail: zhhwang@bjtu.edu.cn

摘 要: 协同过滤方法是当今大多数推荐系统的核心.传统的协同过滤方法专注于评分预测的准确性,然而实际推荐系统的推荐结果往往是项目的排序.针对这一问题,将排名学习领域的知识引入推荐算法,设计了一种基于评分矩阵局部低秩假设的成列协同排名算法.选择直接使用计算复杂度较低的成列损失函数来优化矩阵分解模型,并通过实验验证了其在运算速度上的显著提升.在 3 个实际推荐系统数据集上,与当下主流推荐算法的比较实验结果表明,该算法具有良好的性能.

关键词: 推荐系统;协同过滤;排名学习

中图法分类号: TP311

中文引用格式: 刘海洋,王志海,黄丹,孙艳歌.基于评分矩阵局部低秩假设的成列协同排名算法.软件学报,2015,26(11):2981−2993. http://www.jos.org.cn/1000-9825/4904.htm

英文引用格式: Liu HY, Wang ZH, Huang D, Sun YG. Listwise collaborative ranking based on the assumption of locally low-rank rating matrix. Ruan Jian Xue Bao/Journal of Software, 2015,26(11):2981−2993 (in Chinese). http://www.jos.org.cn/1000-9825/4904.htm

Listwise Collaborative Ranking Based on the Assumption of Locally Low-Rank Rating Matrix

LIU Hai-Yang, WANG Zhi-Hai, HUANG Dan, SUN Yan-Ge

(School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China)

Abstract: Collaborative filtering (CF) is the core of most of today's recommender systems. Conventional CF models focus on the accuracy of predicted ratings, while the actual output of recommender systems is a list of ranked items. In response to this problem, this research introduces technologies in the field of learning to rank into recommendation algorithms and proposes a listed collaborative ranking algorithm based on the assumption that the rating matrix is locally low-rank. It directly uses list-wise ranking loss function to optimize the matrix factorization model. Significant improvement on operation speed is achieved and verified by experiment. Experiments on three real-world recommender system datasets show that the proposed algorithm is a viable approach compared with existing recommendation algorithms.

Key words: recommender system; collaborative filtering; learn to rank

在现代社会,消费者对各种各样的选择应接不暇.电子零售商以及内容提供商为人们提供了广阔的产品选择空间,极大地满足着人们的各种需求.为用户匹配恰当的产品,是提升用户满意度和忠诚度的关键.推荐系统将用户和消费项目紧密连接在一起,为用户提供个性化的推荐,帮助用户在面对大量诸如电影、音乐、图书等产品的时候做出选择[1].当下广泛使用的推荐系统有Netflix面向电影推荐、Amazon.com面向产品推荐、Last.fm, Pandora,iTunes Genius 面向音乐推荐、YouTube 的 Recommended For You 面向在线视频推荐、Facebook 的“你可能认识的人”面向社交网络推荐、What Should I Read Next 面向图书推荐等等.

协同过滤(collaborative filtering,简称 CF)方法是大多数在线商店和社交网络中推荐引擎的核心,在最近的

* 基金项目: 北京市自然科学基金(4142042); 中央高校基本科研业务费专项资金(2015YJS049)
收稿时间: 2015-05-31; 修改时间: 2015-07-14, 2015-08-11; 定稿时间: 2015-08-26

© 中国科学院软件研究所 http://www.jos.org.cn

几十年以来,引起了研究人员的广泛关注.协同过滤方法的主要思想可以理解为:在历史记录中表现出相同喜好的用户,在未来也会偏爱相似的产品[2].首先,协同过滤方法会收集目标用户和其他用户的偏好信息,以此来推断目标用户对所有项目的偏好情况,并根据预测的偏好值对项目进行排序;最终选取排名靠前的 k 个项目作为该用户的推荐结果.

大多数的推荐系统使用评分值来代表用户的偏好,这样,推荐任务就简化为预测能未知的用户-项目评分.为了评价这样的系统的性能,均方误差(mean squared error,简称 MSE)自然成为一种评价指标,具有较小 MSE 值的模型能够做出更好的评分预测.此外,一旦 MSE 被选择作为评价标准,它也自然成为训练模型过程中损失函数的第一选择.尽管这是一种有效且成功的推荐系统构建方法,许多学者依旧认为使用 MSE 作为评价指标对于推荐任务来说是一种欠佳的选择[3].首先,正如之前所提到的,大多数的推荐系统的任务是生成一个项目的 top-k 列表展示给用户.因此,只有系统预测的用户会给予高评分的项目才会被用户看到,用户不会看到那些预测评分低的项目.而 MSE 对所有的评分是同等对待的,最小化模型 MSE 的过程中会同时处理评分低和评分高的项目.这样就会增加许多不必要的工作量,它解决的问题超出了我们所需要解决的问题.其次,通常系统预测的评分值并不会直接显示给用户.如果预测评分值的作用仅仅是用来生成 top-k 列表,那么这些预测值的精确度就显得不是那么重要,而项目的正确排序才是解决问题的关键.

因此,我们希望直接使用排名评价指标来替代 MSE 作为推荐系统模型学习的优化目标.这样,排名学习(learn to rank,简称 LTR)方法[4]就进入了人们的视野.LTR 应用于推荐系统场景的方法也称为协同排名(collaborative ranking,简称 CR)[3].然而,尽管 LTR 技术在信息检索领域取得了很大的进展,其在推荐系统领域的研究还很有限,究其原因主要有两点.第一,推荐系统中的用户和项目难以用精确的属性来表示.反观信息检索应用,其中的“查询”和“文档”可以用出现频率精确表示.推荐系统研究者通常使用用户和项目的评分向量来表示他们,而这样的属性描述方法与用户和项目本身的特性并没有直接的联系.为此,我们选择利用协同过滤中技术的潜在因素作为描述用户和项目的属性.第二,并不是所有的 LTR 方法都可以直接适用于推荐系统场景.Liu 等人[4]将 LTR 方法分为基于点比较(point-wise)、成对比较(pair-wise)、成列比较(list-wise)这 3 种.点比较法为每个“文档”计算一个排名分数,根据这些分数生成排名序列,因此可以将其视为与推荐系统的评分预测相同的概念,如上一段中所述,评分预测的精确度评价指标难以评估排名质量;成对比较法预测的是每个“文档”对在最终排序中的相对顺序,这种方法的计算复杂度较高,在推荐系统的大规模数据场景下难以有高效的表现.为此,我们选择使用 LTR 中的成列比较法.

基于上述问题,我们设计并实现了一种基于评分矩阵局部低秩假设的成列协同排名算法.我们首先假设评分矩阵具有局部低秩性,建立加权化的矩阵分解模型,用以描述用户和项目的属性特征;进而使用排名指标作为其矩阵分解模型的优化目标.在排名指标的选择上,我们选择了一种成列比较排名评价指标——top one probability.这样的方法具有较低的计算复杂度,仅与评分矩阵中的评分个数成线性关系.在实际推荐系统数据集上进行的验证结果显示,我们的算法是准确而有效的.

本文第 1 节综述相关研究内容.第 2 节详细阐述我们的算法及其优化过程.相应的算法验证实验方式和实验结果将在第 3 节中给出.最后,在第 4 节中对全文做出总结与展望.

1 相关研究综述

我们的算法结合了协同过滤以及排名学习两个领域的技术,下面我们从这两个方面综述以往相关的研究内容.

1.1 协同过滤

协同过滤方法是现今最成功的推荐技术之一.与基于内容的推荐算法相比,协同过滤方法无需相关领域的专业知识,只依赖于用户的历史行为记录,例如他们的交易记录或对产品的评价,从而避免了额外的数据收集需求.因此在过去的几十年间,协同过滤技术引起了广泛的关注,取得了显著的进展,并被应用于一些成功的商业系统,如 Amazon[5],Tivo 和 Netflix.

© 中国科学院软件研究所 http://www.jos.org.cn

协同过滤方法通常可以分为两类:基于历史行为(memory-based)的方法和基于模型(model-based)的方法.

基于历史行为的协同过滤方法也被称为基于近邻(neighborhood-based)的协同过滤算法.这种方法的核心是计算用户或项目之间的相关性,常见的步骤是:首先计算两两用户(或项目)间的相似度,这里的相似度体现了用户(或项目)之间的距离、相关性或是权重;然后,根据系统用户(或项目)对于某一项目(或用户)的所有评分的加权平均数得到用户对该项目的预测评分.

最初的近邻模型协同过滤算法是基于用户的(user-based).这种方法依据相似用户的评分记录来预测未知评分[6,7].此后,基于项目的(item-based)方法逐渐变得流行[5,8].与基于用户的方法类似,基于项目的方法寻找与目标项目相似的项目集合,根据这些相似项目的评分得到预测评分.更好的可扩展性和更高的精确度,使得基于项目的方法在很多场景中取得了更好的效果[8,9].除此之外,基于项目的方法对于预测结果具有更好的解释性,这是因为用户往往对于他们之前购买过的项目更为熟悉,却并不熟识与他们志趣相投的用户.

协同过滤中的另一类算法是基于模型的算法.基于模型的算法根据训练数据建立参数模型,依照学习到的模型预测未知评分并生成推荐结果.很多机器学习方法和数据挖掘技术都可以被用来建立推荐模型.Miyahara 和 Pazzani 将朴素贝叶斯模型应用于协同过滤任务中[10,11],将推荐问题转化为分类问题.Sarwar 等人使用聚类技术把评分数据进行聚类,再利用近邻模型进行预测[12].Vucetic 和 Obradovic 提出了一种基于回归的协同过滤方法,通过建立一系列线性模型来有效预测用户评分[13].slope-one 算法运用更简单形式的回归表达式和单一的自由参数,提供了一种快速且高效的协同过滤方法[14].其他的一些例子还包括使用潜在语义模型分析(probabilistic latent semantic analysis,简称 pLSA)的模型[15]、使用神经网络的模型[16]等.

近年来,一类基于矩阵分解(matrix factorization,简称 MF)的协同过滤方法开始变得流行.矩阵分解方法将评分矩阵分解为两个低秩矩阵(用户矩阵和项目矩阵)的乘积,利用这两个矩阵来预测原始评分矩阵中的缺损值.总体上来讲,矩阵分解方法是在给定损失函数和正则项的前提下的优化问题,不同的损失函数和正则项构成不同的矩阵分解模型.这里,我们将具有 m 个用户和 n 个项目的评分矩阵表示为 $M\in\mathbb{R}^{m\times n}$,为了给用户推荐新的项目,需要预测矩阵 M 中的未知项,在 M 是低秩矩阵的假设条件下,以平方损失函数作为优化目标,通常的矩阵分解方法可以通过解决公式(1)中的优化问题来达到这一目的:

$$(\hat{U},\hat{V})=\arg\min_{U,V}\sum_{(u,i\in A)}(M_{u,i}-[UV^T]_{u,i})^2 \tag{1}$$

其中,$U\in\mathbb{R}^{m\times r}$,$V\in\mathbb{R}^{n\times r}$($r$ 远远小于 m 和 n),A 是从矩阵 M 内已知评分中选取的训练集合.由此,我们可以得到近似矩阵 $\hat{M}=\hat{U}\hat{V}^T$,它可以用来估计 $M_{v,j}$ 的值,包括$(i,j)\notin A$ 的情况.这样,对于某一用户 v,根据从集合 $\{\hat{M}_{v,j}:(v,j)\notin A\}$ 中选取的 k 个最大值,就可以得到该用户的推荐项目列表.

常见的矩阵分解算法有规范化的 SVD(regularized SVD)[17]、非负矩阵分解(non-negative matrix factorization,简称 NMF)[18]、概率矩阵分解(probabilistic matrix factorization,简称 PMF)[19]、贝叶斯概率矩阵分解(Bayesian PMF)[20]、非线性概率矩阵分解(non-linear PMF)[21]、最大边界矩阵分解(maximum margin matrix factorization,简称 MMMF)[22]、非线性主成分分析(nonlinear principal component analysis,简称 NPCA)[23]、SVD++[24]等.

1.2 协同排名

排名学习已经成为机器学习和信息检索领域的重要研究问题.以标签排名问题为例,设 X 为一个实例集合,Y 为一组标签的集合,某一实例 $x\in X$ 的标签排名问题需要对 Y 中的标签进行全排序,其中,$y_1\succ y_2$ 表示用户 x 比 y_1 更偏爱 y_2.排名函数 $f:X\times Y\to\mathbb{R}$返回一个数值,表示标签 y 与实例 x 的相关程度.我们需要学习函数 f,使其与训练集合中的偏好顺序尽量一致.也就是说,给定一个 x,对于每个 $y_1\succ y_2$ 关系,要让 f 尽量满足 $f(x,y_1)>f(x,y_2)$.

在推荐系统领域,U 代表的是用户集合$\mathcal{U}$,I 代表项目集合$\mathcal{I}$.函数 $f(u,i)$用来估计用户 $u\in\mathcal{U}$对于项目 $i\in\mathcal{I}$的偏好得分,如果用户 u 对于项目 i_1 的喜好强于项目 i_2,则 f 需满足 $f(u,i_1)>f(u,i_2)$.

大多数推荐系统是以一列排序好的项目作为最终的输出形式的,从这个角度来看,从预测评分得到项目的

© 中国科学院软件研究所 http://www.jos.org.cn

排名比预测评分本身显得更为重要.在某些应用场景下,用户甚至无法通过评分来表达其对于项目的偏好程度,系统中只有例如点击记录、下载记录一类的用户行为信息.这类相对不明显的反馈信息,仅仅表明用户有可能会喜欢哪些项目.这就意味着,传统的基于评分预测的协同过滤算法无法适用于这种反馈.

这里,我们通过两个例子来说明基于评分的协同过滤与基于排名的协同过滤的区别和相互之间的关系.在第 1 个例子中,我们假设某用户分别对项目 i 和 j 给予 4 分和 3 分的项目评分,如果有两个推荐系统对项目(i,j)的预测评分分别是(3,4)和(5,2),根据评分预测的标准,如平均绝对误差(mean absolute error,简称 MAE)或均方根误差(root mean squared error,简称 RMSE),对这两个推荐结果的评价将会是相同的,只有从排名的角度才能看出预测评分(5,2)可以真实反映出该用户更偏好的项目 i.然而,这并不能说明基于评分的算法对推荐系统性能有不好的影响,恰恰相反,现今大多数成功的推荐系统还是利用基于评分的算法来生成推荐列表的.在第 2 个例子中,如果我们得到用户 u 和 v 对于项目(i,j)的评分是(5,3)和(4,3),则可知用户 u 对于项目 i 的偏爱更加明显,这样的评分信息可以增强生成推荐列表的确信度.此外,预测评分值可以为用户提供额外的信息来帮助他们决定是否点击、购买或下载所推荐的项目.综合以上两个例子,无论是基于评分还是基于排名的协同过滤算法,都对最终的推荐列表生成发挥着自己的作用,同时,二者还可以互相弥补彼此的缺陷.这启发着我们结合这两种方法,设计出成功的推荐算法.

基于排名的协同过滤算法也被称为协同排名(collaborative ranking),与大量基于评分的协同过滤算法相比,协同排名算法的研究数量目前还很有限.与信息检索领域不同,推荐系统领域的排名问题的困难在于,难以找到并表示用户和项目的显性特征.因此,文献[25]建立用户的 k 近邻空间,并从中抽取(u,i)对的特征,而后将(u,i)对的特征带入排名学习算法中,求解排名问题.还有一些协同排名算法是基于排名学习和矩阵分解的.Ordrec[26]是一种基于序数回归和 SVD++的排名算法,它通过 SVD++对用户评分建模,将最小化序数回归损失作为优化目标.该算法的优势在于:能够估计每个预测值的置信度,并且可以处理非数值型用户反馈.文献[27]为个性化排名提出了一种一般性的优化标准 Bpropt,即,从贝叶斯方法的角度推导排名问题,并最大化其后验概率.Bpropt 基于矩阵分解和 KNN,优化目标为 ROC 曲线下面积(area under the ROC curve,简称 ROC),使用梯度下降和 bootstrap 抽样方法来更新参数.PMF-co[3]使用成对比较法来解决推荐问题.在算法学习过程中,需同时优化用户-项目特征和排名函数的权重.ListRankMF[28]的目标在于最小化预测项目排序和真实排序之间的互熵.文献[29]基于 Bradley-Terry 模型的混合分布对用户偏好直接建模,提出了一种概率潜在偏好分析模型.此外,一些协同排名算法直接对排名学习的评价指标进行优化.TFMAP[30]使用张量分解来对显性反馈数据和上下文信息建模,并直接将最大化平均准确率作为优化目标.CLiMF[31]通过最大化平均排序倒数(mean reciprocal rank)来学习模型参数,CofiRank[32,33]建立了一个最大边际矩阵分解模型,以最小化 NDCG 的上边际.

Lee 等人在文献[34]中提出了一种局部协同排名(local collaborative ranking,简称 LCR)算法.这也是一种矩阵分解算法和 LTR 算法结合的算法.在矩阵分解阶段,他们假设评分矩阵具有局部低秩性,为(u,i)构造近邻空间,在近邻空间内进行矩阵分解计算.在重建阶段,LCR 选择使用排名损失函数中的 0-1 损失函数,经过梯度下降过程达到损失最小的优化目标.LCR 算法中存在着一些问题:首先,推荐系统的实际任务是要预测正确的项目排序,而 LCR 所选择的 0-1 损失函数的目标是最小化项目对之间的误差,而非整个项目排序的误差;其次,0-1 损失函数是一种成对比较法(pair-wise),它需要从单一评分中获取成对训练实例,具有较高的计算复杂度,其计算量随有评分项的个数呈二次方增长,这限制了 LCR 算法的扩展性.如今,现实中的推荐系统往往都是在大规模数据背景下的,因此在排名学习阶段,我们选择使用成列比较法(list-wise),直接考虑整个项目排序,同时还具有较低的计算复杂度.

2 基于评分矩阵局部低秩假设的成列比较协同排名算法

在本节中,我们提出一种基于评分矩阵局部低秩假设的成列比较协同排名算法.首先,我们将介绍该算法的两个关键组成部分:局部低秩矩阵分解和成列排名损失函数.在此基础上,我们将阐述我们的算法,并探讨其在优化目标下的学习过程.

© 中国科学院软件研究所 http://www.jos.org.cn

2.1 局部低秩矩阵分解

首先,我们选择利用协同过滤中技术的潜在因素作为描述用户和项目的属性.如前文所述,基于潜在因素模型的算法有很多成功的例子,这里,我们选择使用 Lee 提出的局部低秩矩阵分解法(local low-rank matrix approximation,简称 LLORMA)[35].

局部低秩矩阵分解假设评分矩阵(行,列)对空间$\Phi=\{(u,i):u=1,\ldots,m,i=1,\ldots,n\}$中具有一个函数 d 衡量着用户-项目对之间的距离.根据距离函数 d,可以得到用户-项目对的近邻,局部低秩假设指出,矩阵 M 可以用一组低秩矩阵 $\hat{M}_s$ $(s\in\Phi)$来近似估计,其中,每个矩阵 $\hat{M}_s$ 都是低秩矩阵,它们在 s 的近邻空间 $N_s=\{s'\in\Phi:d(s,s')<\alpha\}$内局部地描述矩阵 M.

矩阵局部低秩假设思想来源于这样的思考:推荐系统中存在着一些相似的用户-项目对,例如儿童用户观看卡通电影,我们假设这样的一些子矩阵具有低秩性,而非评分矩阵 M 从整体上具有低秩性.因此,近邻 N_s 的定义关注的用户-项目对之间的内在联系,与用户和项目的坐标无关,即,距离函数 $d((a,b),(c,d))$不会因$|c-a|$和$|d-b|$小而小.

LLORMA 首先随机选择以 q 个锚点 $s_1,\ldots,s_q\in\Phi$为中心的 q 个近邻空间,然后,通过最小化重建平方差,在每个近邻内空间计算一个低秩矩阵,如公式(2)所示:

$$\arg\min_{U,V}\sum_{(u,i\in A)}K((u^*,i^*),(u,i))([UV^T]_{u,i}-M_{u,i})^2 \tag{2}$$

其中,$K((u^*,i^*),(u,i))$是一个二维核平滑(smoothing kernel)函数,用来衡量重构点(u,i)和锚点(u^*,i^*)之间的临近程度.LLORMA 中使用两个核函数 $K(u^*,i^*)$,$K(u,i)$的乘积来实现这一目的.上式中的优化问题从本质上来说是传统 SVD 优化问题的加权式,只是它需要重复 q 次.

在实际应用中,不可能对每个用户-项目对计算一个上式中的加权 SVD 问题.所以,LLORMA 算法需要事先在训练集中选取 q 个锚点,而不是将每个用户-项目对作为锚点.由 q 个锚点定义的 q 个局部模型通过线性组合最终得到预测评分:

$$\hat{M}_{u,i}=\sum_{t=1}^{q}\frac{K((u_t,i_t),(u,i))}{\sum_{s=1}^{q}K((u_s,i_s),(u,i))}[\hat{U}_t\hat{V}_t^T]_{u,i} \tag{3}$$

其中,(u_t,i_t)是局部模型 t 中的锚点.公式(3)可以看作是每个局部预测模型的凸组合,它保证了距离待估用户-项目对(u,i)越近的模型在计算中的贡献越大.

LLORMA 算法以矩阵局部低秩性假设替代了传统的整体低秩性假设,在实验中表现出了较好的预测准确性.从文献[35]中的实验结果(见表 1)可以看出,随着矩阵秩数的增加,SVD 和 LLORMA 算法的性能均会随之提升;而且在每一个给定的秩数下,LLORMA 算法的 RSME 值均小于 SVD 算法.

Table 1 RMSE of SVD and LLORMA on three datasets

表 1 传统 SVD 算法和 LLORMA 算法在 3 个数据集上 RSME 值的比较

Rank	MovieLens (1M)		MovieLens (10M)		Netflix	
	SVD	LLORMA	SVD	LLORMA	SVD	LLORMA
1	0.920 1	0.913 5	0.872 3	0.865 0	0.938 8	0.929 5
5	0.873 7	0.853 7	0.825 5	0.804 9	0.883 6	0.860 4
10	0.865 0	0.839 6	0.821 9	0.788 9	0.876 5	0.844 4
20	0.864 7	0.833 3	0.822 0	0.781 5	0.874 2	0.833 7

2.2 成列排名损失函数

由于大多数推荐系统是以一列排序好的项目作为最终的输出形式的,在提出了基于评分预测的 LLORMA 算法之后,原作者也将其应用到了排名学习的场景中,提出了上一节中提到的 LCR 算法.然而,由于 LCR 选择的是成对比较排名损失函数,算法存在着计算复杂度高的弊端.在此,我们选择一种成列排名损失函数——top one probability.

如文献[36]中所述,从概率的角度来讲,top one probability 表示一个评分项在所有评分项中排名第一的概

© 中国科学院软件研究所 http://www.jos.org.cn

率.具体而言,评分为 R_{ui} 的项目 i 在用户 u 的项目排名中(假设排名中共有 n 个项目)的 top one probability 可以表示为

$$P(R_{ui})=\frac{\varphi(R_{ui})}{\sum_{k=1}^{n}\varphi(R_{uk})} \tag{4}$$

其中,$\varphi(x)$可以是任意的单调递增严格正函数.为了简便起见,我们使用和文献[36]一样的方法,选择指数函数作为$\varphi(x)$.这样,公式(4)就变成:

$$P(R_{ui})=\frac{\exp(R_{ui})}{\sum_{k=1}^{n}\exp(R_{uk})} \tag{5}$$

2.3 基于评分矩阵局部低秩假设的成列协同排名算法

相对于传统推荐算法中预测评分的做法,我们的思路是利用矩阵分解模型去学习每个用户的项目排序.我们将矩阵分解模型中的项目排序与训练集中的真实项目排序进行比较,将二者 top one probability 的互熵作为损失函数,公式表示为

$$\begin{aligned}L(U,V)&=\sum_{u=1}^{m}\left\{-\sum_{i=1}^{n}I_{ui}P(R_{ui})\log P(g(U_u^TV_i))\right\}+\frac{\lambda}{2}(\|U\|_F^2+\|V\|_F^2)\\&=\sum_{u=1}^{m}\left\{-\sum_{i=1}^{n}I_{ui}\frac{\exp(R_{ui})}{\sum_{k=1}^{n}I_{uk}\exp(R_{uk})}\log\frac{\exp(g(U_u^TV_i))}{\sum_{k=1}^{n}I_{uk}\exp(g(U_u^TV_i))}\right\}+\frac{\lambda}{2}(\|U\|_F^2+\|V\|_F^2)\end{aligned} \tag{6}$$

其中,I_{ui}为指示函数:当$R_{ui}>0$时,$I_{ui}=1$;其他情况下,$I_{ui}=0$.$\|U\|_F^2$和$\|V\|_F^2$为矩阵的弗罗贝尼乌斯范数,λ是为防止过拟合现象而设置的正则化系数.为简便起见,我们为$\|U\|_F^2$和$\|V\|_F^2$设置了相等的正则化系数.这样,一个损失函数反映了使用训练排序训练矩阵分解模型所生成的预测排序的不确定性,通过最小化这个规范化的损失函数,可以得到矩阵分解模型的结果 U,V.

我们选择使用梯度下降法交替更新 U 和 V 来达到寻求局部最小值.在 $L(U,V)$中,U,V 的梯度计算如下:

$$\frac{\partial L}{\partial U_u}=\sum_{i=1}^{n}I_{ui}\left(\frac{\exp(g(U_uV_i^T))}{\sum_{k=1}^{n}I_{ik}\exp(g(U_uV_i^T))}-\frac{\exp(R_{ui})}{\sum_{k=1}^{n}I_{ik}\exp(R_{ik})}\right)g'(U_uV_i^T)V_i+\lambda U_u \tag{7}$$

$$\frac{\partial L}{\partial V_i}=\sum_{u=1}^{m}I_{ui}\left(\frac{\exp(g(U_uV_i^T))}{\sum_{k=1}^{n}I_{ik}\exp(g(U_uV_i^T))}-\frac{\exp(R_{ui})}{\sum_{k=1}^{n}I_{ik}\exp(R_{ik})}\right)g'(U_uV_i^T)U_u+\lambda V_i \tag{8}$$

其中,$g(U_i^TV_j)$ 即公式(3)中的局部矩阵分解模型加权后的结果.进一步来说,根据公式(3),我们有:

$$\frac{\partial g}{\partial[U_t]_{u,k}}=[V_t]_{i,k} \tag{9}$$

$$\frac{\partial g}{\partial[V_t]_{i,k}}=[U_t]_{u,k} \tag{10}$$

我们将我们的算法命名为 LLCR 算法(listed local collaborative ranking).LLCR 算法学习过程的伪代码如算法 1 所示.

算法 1. LLCR 算法学习过程.

输入:评分矩阵 $M\in\mathbb{R}^{m\times n}$.

参数:局部模型个数 q,局部矩阵的秩 r,学习速率 ν,正则化参数λ.

1: 定义:A 是从矩阵 M 内已知评分中选取的训练集合

2: for all $t\in\{1,\ldots,q\}$ do

3: 使用随机数值初始化 $U_t\in\mathbb{R}^{m\times r}$,$V_t\in\mathbb{R}^{n\times r}$.

4: 随机从 A 中选择(u_t,i_t)对

5: end for

© 中国科学院软件研究所 http://www.jos.org.cn

while not-converged do
  for all $(u,i)\in A$ do
    $\omega_{u,i} \leftarrow \sum_{t=1}^{q} K(u_t,u)K(i_t,i)$
    $f_{u,i} = \sum_{t=1}^{q} \frac{K(u_t,u)K(i_t,i)}{\omega_{u,i}}[U_t V_t^T]_{u,i}$
    $\ell_{u,i} = \frac{\partial L}{\partial f_{u,i}}$
  end for
  for all $t\in\{1,\ldots,q\}$ do
    $\forall u\in\{1,\ldots,m\}:[\triangle U]_u \leftarrow 0$
    $\forall i\in\{1,\ldots,m\}:[\triangle U]_i \leftarrow 0$
    for all $t\in\{1,\ldots,q\}$ do
      $[\triangle U]_u \leftarrow [\triangle U]_u + \frac{K(u_t,u)K(i_t,i)}{\omega_{u,i}}\cdot[V_t]_i\cdot\ell_{u,i}$
      $[\triangle V]_i \leftarrow [\triangle V]_i + \frac{K(u_t,u)K(i_t,i)}{\omega_{u,i}}\cdot[U_t]_u\cdot\ell_{u,i}$
    end for
    $\forall u\in\{1,\ldots,m\}:[U_t]_u \leftarrow [U_t]_u - \nu\left(\frac{[\triangle U]_u}{s_u} + \lambda[U_t]_u\right)$
    $\forall i\in\{1,\ldots,n\}:[V_t]_i \leftarrow [V_t]_i - \nu\left(\frac{[\triangle V]_i}{s_u} + \lambda[V_t]_i\right)$
  end for
end while
输出:$U_t V_t^T, t\in\{1,\ldots,q\}$

LLCR 算法对未知测试实例(u^*,i^*)的预测过程的伪代码如算法 2 所示.

算法 2. LLCR 算法预测过程.

输入:已学习到的局部矩阵模型 $U_t V_t^T$ $(t\in\{1,\ldots,q\})$,未知评分项(u^*,i^*).

Return

$$\hat{M}_{u,i} = \sum_{t=1}^{q} \frac{K((u_t,i_t),(u,i))}{\sum_{s=1}^{q} K((u_s,i_s),(u,i))}[\hat{U}_t \hat{T}_t^T]_{u,i}$$

2.4 计算复杂度分析

考虑到数据集的稀疏性,公式(6)中的损失函数的计算复杂度为 $O(2d|A|+d(m+n))$;公式(7)和(8)中梯度计算的计算复杂度分别为 $O(2d|A|+dm)$和 $O(d|A|+pd|A|+dn)$,其中,p 表示每个用户平均评价过的项目个数,其大小通常要比$|A|$小得多.考虑到实际推荐系统数据中$|A|$要远远大于 m 和 n,则我们算法一次循环的总体计算复杂度为 $O(d|A|+pd|A|)$,与矩阵中已有评分的个数成线性相关;而如 LCR 一样的基于成对比较法的算法的计算量是随矩阵中已有评分的个数呈二次方增长的.这样的计算复杂度保证了我们的算法具有计算上的高效性,并且可以应用于大规模数据推荐场景.

3 实 验

我们选择借助 Prea(http://prea.gatech.edu/index.html)平台来实现并验证我们的算法[37].Prea(personalized recommendation algorithms toolkit)平台实现了大量的推荐系统算法,包括时下主流的算法以及传统的经典算

© 中国科学院软件研究所 http://www.jos.org.cn

法.此外,它还提供了多种常用的评价指标及数据集.Prea 平台可被用于进行多个不同推荐算法之间的公平且全面的比较实验.其评价流程、数据接口、开源等特性使得在其基础上进行新算法的实现过程显得十分便捷.

3.1 数据集合

我们选择了 3 个实际推荐系统中的数据集来验证 LLCR 算法.其相关统计数据见表 2.

Table 2 Statistical information of Datasets used in the experiments

表 2 实验数据集统计信息

Dataset	User	Item	Rating	Density
MovieLens 100K	943	1 682	100 000	6.3%
MovieLens 1M	6 039	3 883	1 000 209	4.3%
Netflix	4 427	1 000	56 136	1.3%
EachMovie	18	21%	21%	2.8%

3.1.1 MovieLens 数据集(http://www.grouplens.org)

明尼苏达大学计算机科学与工程系的 GroupLens Research 实验室于 1997 年开始研究与开发称为 MovieLens 的项目,该项目提供了关于电影的一种推荐系统和虚拟社区网站,其本身目的是:利用该网站所搜集的数据,研究如何构建更好的推荐系统.具体研究内容包括推荐系统、在线社区、移动与普适技术、数字图书馆、和本地地理信息系统.GroupLens Research 实验室建立了 MovieLens 网站.该网站包括了大量的电影资源,用户可以对电影进行评分或贴标签.通过对用户的历史记录分析和相似用户的行为分析,该网站可以向用户推荐其可能感兴趣的电影.GroupLens Research 实验室从该网站上收集并整理了一些用于推荐系统等研究的数据集合,这些数据集合是在不同时期进行收集的,数据集合的大小也不相同.

我们选择的是 MovieLens 100K 数据集,其中包含 943 个用户对 1 682 部电影所做的 100 000 个(即 100K 个)评分.这样一个比较小的数据集适用于进行快速的算法基准测试,我们将使用它来研究我们算法中的参数特性.

我们还选择了 MovieLens 1M 数据集用于算法的横向比较实验,其中包含 6 039 个用户对 3 883 部电影所做的 1 000 209 个(即 1M 个)评分.

3.1.2 Netflix 数据集

Netflix 是一家在线影片租赁提供商,公司能够提供超大数量的 DVD,而且能够让顾客快速、方便地挑选影片,同时免费递送.Netflix 大奖赛从 2006 年 10 月开始,Netflix 公开了大约 1 亿个 1~5 的匿名影片评级,数据集仅包含了影片名称、评价星级和评级日期,没有任何文本评价的内容.比赛要求参赛者预测 Netflix 的客户分别喜欢什么影片,要把预测的准确率提高 10%以上.我们从原始的数据集中抽取了 4 427 个用户、1 000 个项目、56 136 个评分作为实验数据集.

3.1.3 EachMovie 数据集

HP/Compaq 的 DEC 研究中心曾经在网上架设 EachMovie 电影推荐系统对公众开放.之后,这个推荐系统关闭了一段时间,其数据作为研究用途对外公布,MovieLens 的部分数据就来自于这个数据集.这个数据集有 72 916 个用户对 1 628 部电影进行的 2 811 983 次评分.早期大量的协同过滤的研究工作都是基于这个数据集的.

3.1.4 数据分割

按照参考文献[34]中的设定,我们以两种方式对数据集进行分割:

- 在第 1 部分的实验中,我们使用固定比率分割法来研究 LLCR 算法.即,在训练集中为每个用户选择一个固定比例的评分个数,将该用户其余的评分项置于测试集中;
- 在第 2 部分的实验中,我们选择固定评分个数来进行 LLCR 算法与其他推荐算法的比较.即,在训练集中为每个用户随机选择固定数量的评分,将剩余所有的评分置于测试集中,没有足够评分的用户将从测试集中移除.

3.2 评价指标

我们还选择了 3 种排名评价标准来验证我们的算法:

© 中国科学院软件研究所 http://www.jos.org.cn

- 1 损失(zero-one error)衡量的是排序中每对项目相对顺序的平均正确率,公式表示为

$$\varepsilon_{0/1}=\sum_{u\in U}\sum_{i\in T_u}\sum_{j\in M_u\cup T_u\setminus\{i\}}\frac{1}{Z_u}1[\Delta M_{u,i,j}\cdot g(u,i,j)<0],$$

其中,$Z_u=|U|\cdot|T_u|\cdot(|M_u\cup T_u|-1)$,$M_u$ 和 T_u 分别表示用户 u 在训练集和测试集中的已评分集合.

- 平均准确率(average precision)的定义为准确率/召回率曲线下的区域面积,公式表示为

$$AvgP=\int_0^1 P(r)\mathrm{d}r,$$

其中,积分变量 r 覆盖所有可能的召回率等级,$P(r)$表示在召回率 r 下的准确率.在实际计算中,使用推荐序列中每个位置进行求和来代替积分运算,即

$$AvgP=\frac{1}{|U|}\sum_{u\in U}P(u)\Delta r(u).$$

- DCG(discounted cumulative gain)是一种用于衡量搜索引擎质量的指标.

在这种评价方法中,每一个“文档”都对它所在的位置有一定的贡献,其贡献值与“文档”的相关度有关.然后,1~n 的所有位置上的贡献值都被加起来作为最终的评价结果.这样,一个一定长度的文档序列被转换成了一个相关分值的序列.给定一个排序后的文档序列,在第 k 位的 NDCG(normalized discounted cumulative gain)值 $DCG@k$ 的计算公式为

$$DCG@k=\sum_{i=1}^{k}\frac{2^{rel_i}-1}{\log_2(i+1)},$$

其中,i 表示项目在推荐列表中的位置,rel_i 表示第 i 个项目与用户的相关度.这里,我们采用评分值来衡量项目与用户之间的相关程度.NDCG 是用户 DCG 值与可能的最大 DCG 值之间的比率.当推荐列表中的项目恰好按照用户的偏好程度降序排列时,DCG 取得最大值.NDCG 反映了将合适的项目排在序列前部的重要性,已经成为最常用的排序评价指标之一.在我们的实验室中,我们选择使用:

$$NDCG@k=10.$$

4 结果分析

首先,我们通过实验比较了 LLCR 算法和原始 LCR 算法的运算时间.实验中,我们使用的是一台 Intel(R) Core(TM) i5-3450 3.40GHz 处理器 8GB 内存的计算机.我们选择 Movielens 100K 数据集,并将其随机划分为 50%的训练集和 50%的测试集.我们针对不同的局部矩阵个数(model)和矩阵秩数(rank)进行两个算法训练和测试时间的比较实验,实验中,两个算法的全部参数均设为相同的值.最终实验结果见表 3.

Table 3 Comparison of runtime between LLCR and LCR

表 3 LLCR 算法与 LCR 算法运算时间比较

Model	Rank	Algorithm	Train time	Test time
5	1	LCR	00:00:25.518	00:00:03.440
		LLCR	00:00:11.993	00:00:03.485
5	5	LCR	00:04:08.769	00:00:03.434
		LLCR	00:00:52.498	00:00:03.541
5	10	LCR	00:12:19.792	00:00:03.727
		LLCR	00:02:02.785	00:00:03.739
10	1	LCR	00:00:46.393	00:00:03.462
		LLCR	00:00:22.215	00:00:03.562
10	5	LCR	00:13:32.455	00:00:03.979
		LLCR	00:03:03.268	00:00:04.096
10	10	LCR	00:26:49.845	00:00:04.474
		LLCR	00:07:04.379	00:00:04.611

从实验结果中可以看出,无论在哪种条件下,我们的 LLCR 算法的训练时间都要明显少于 LCR 算法;而且随着局部矩阵个数和矩阵秩数的增加,LLCR 算法的速度优势愈发明显.这样的实验结果符合我们前文计算复杂度分析的结论,说明 LLCR 算法具有更快的运算速度,可以更好地适用于大规模数据推荐场景.

© 中国科学院软件研究所 http://www.jos.org.cn

接下来将 LLCR 算法推荐性能的实验分为两大部分:

- 在第 1 部分的实验中,我们主要研究 LLCR 算法内局部矩阵个数及局部矩阵的秩对算法性能的影响.

这里,我们选择使用 MovieLens 100K 数据集,首先固定局部矩阵模型个数为 10,取局部矩阵的秩数为{10,15,20};而后固定局部矩阵的秩数为 15,取局部矩阵模型个数为{10,20,50}.在核平滑函数的选择中,我们使用 Epanechnikov 核函数,设置其带宽为 0.8.实验结果如图 1~图 6 所示,图中横轴数字表示迭代次数.

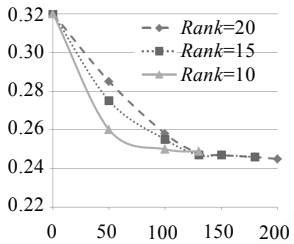


Fig.1 Effect of the model rank on the performance of zero one error

图 1 局部矩阵的秩数对 0-1 损失的影响

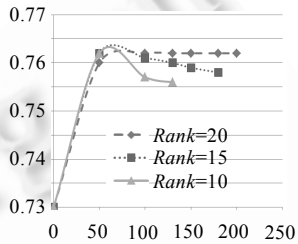


Fig.2 Effect of the model rank on the performance of average precision

图 2 局部矩阵的秩数对平均准确率的影响

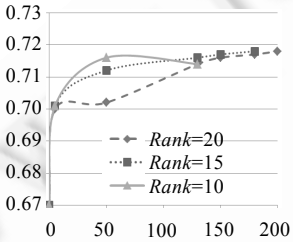


Fig.3 Effect of the model rank on the performance of *NDCG*@10

图 3 局部矩阵的秩数对 *NDCG*@10 的影响

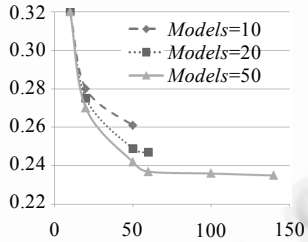


Fig.4 Effect of the number of local models on the performance of zero one error

图 4 局部矩阵的个数对 0-1 损失的影响

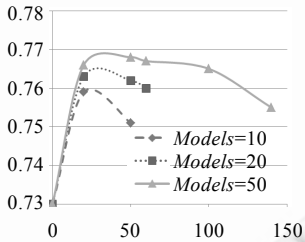


Fig.5 Effect of the number of local models on the performance of average precision

图 5 局部矩阵的个数对平均准确率的影响

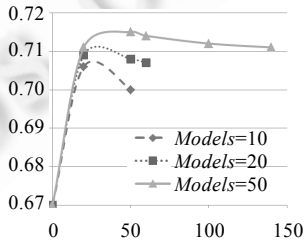


Fig.6 Effect of the number of local models on the performance of *NDCG*@10

图 6 局部矩阵的个数对 *NDCG*@10 的影响

从图 1~图 3 中我们可以看出,随着局部矩阵秩数的增加,0-1 损失的结果都会越来越好,同时也需要更多的迭代次数以达到收敛;而平均准确率和 *NDCG*@*k*=10 的结果却表现出了不同的趋势,它们会在某一点出现过拟合,也就是说,我们需要进行交叉验证以使迭代计算提前结束.但当秩数继续增加时,过拟合现象的影响就会减小.3 项指标总体来看,当 *rank*=10 时,算法的性能无法达到理想的效果;当 *rank*=20 时,虽然评价指标往往可以达到最佳效果,但是通常需要消耗大量的迭代次数;而当 *rank*=15 时,算法仅需要较少的迭代次数即可获得不明显差于 *rank*=20 的评价指标.因此,我们得出的结论是:算法选择局部矩阵秩数为 15 左右、循环次数控制在 100 次

© 中国科学院软件研究所 http://www.jos.org.cn

左右最为理想.

从图 4 中我们可以看出,随着局部矩阵模型个数的增加,0-1 损失的值逐渐减小,迭代次数逐渐增加.图 5、图 6 显示,平均准确率和 *NDCG@k*=10 会出现过拟合现象,但随着局部矩阵个数的增加,过拟合的影响有减小的趋势.总体来看,使用更多的局部矩阵 LLCR 算法会取得更好的性能;但随着局部矩阵个数的提升,所带来的性能上的提高有越来越不明显的趋势.因此我们认为,在局部矩阵个数的选择上,50 是一个较为合适的选择;同时,将迭代次数控制在 50 左右.这样不仅可以得到较为理想的实验结果,而且基本避免了过拟合现象的出现.

- 在第 2 部分的实验中,我们将进行 LLCR 算法与时下主流推荐算法的比较.

我们选择用于比较的算法有基于矩阵分解的算法 regularized SVD[17],NMF[18],PMF[19]以及基于排名学习的算法 cofirank[32],LCR[34].

在参数设置上,我们设 LLCR 的局部矩阵个数为 50,局部矩阵的秩为 15.使用 Epanechnikov 核函数作为核平滑函数,设置其带宽为 0.8.我们选择 EachMovie,Movielens 1M 和部分 Netflix 这 3 个数据集来进行比较实验.

从表 4~表 6 的实验结果中可以看出,与其他算法相比,LLCR 算法和 LCR 算法在两项评价标准上都有着较为明显的优势.而且根据我们前文的分析,LLCR 与 LCR 相比具有更低的计算复杂度,在面对更大规模的推荐系统数据时,LLCR 比 LCR 有更强的适用性.

Table 4 Experimental results on EachMovie dataset

表 4 EachMovie 数据集实验结果

Algorithm	*NDCG@k*=10	Average precision
SVD	0.698	0.731
NMF	0.639	0.677
PMF	0.674	0.717
CofiRank	**0.716**	0.723
LCR	0.699	**0.733**
LLCR	0.701	0.725

Table 5 Experimental results on MovieLens 1M dataset

表 5 MovieLens 1M 数据集实验结果

Algorithm	*NDCG@k*=10	Average precision
SVD	0.678	0.750
NMF	0.647	0.733
PMF	0.596	0.652
CofiRank	0.691	0.692
LCR	0.702	**0.763**
LLCR	**0.707**	0.758

Table 6 Experimental results on Netflix dataset

表 6 Netflix 数据集实验结果

Algorithm	*NDCG@k*=10	Average precision
SVD	0.777	0.682
NMF	0.775	0.671
PMF	0.704	0.584
CofiRank	0.724	0.668
LCR	0.779	**0.721**
LLCR	**0.784**	0.713

5 总 结

在本文中,针对推荐系统中的排名问题,我们设计并实现了一种基于评分矩阵局部低秩假设的成列协同排名算法——LLCR 算法.该算法结合了协同过滤技术和排名学习技术两个领域的内容,首先,我们假设评分矩阵具有局部低秩性,进而选择使用成列排名函数——top one probability 来优化其矩阵分解模型.经实验验证:我们的算法与成对比较排名算法相比在运算速度上有着明显的提升;同时,在 NDCG、平均准确率等各项评价指标

© 中国科学院软件研究所 http://www.jos.org.cn

上展现了良好的性能.因此,LLCR 算法在保证推荐结果质量的同时,可以更好地适用于大规模数据推荐场景.

致谢 在此,我们向对本文的工作给予支持和建议的同行,特别是对北京交通大学计算机与信息技术学院的王志海教授表示感谢.

References:

[1] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. on Knowledge and Data Engineering, 2005,17(6):734−749. [doi: 10.1109/TKDE.2005.99]

[2] Su X, Khoshgoftaar TM. A survey of collaborative filtering techniques. In: Proc. of the Advances in artificial intelligence. 2009. [doi: 10.1155/2009/421425]

[3] Balakrishnan S, Chopra S. Collaborative ranking. In: Proc. of the 5th ACM Int'l Conf. on Web Search and Data Mining. ACM Press, 2012. 143−152. [doi: 10.1145/2124295.2124314]

[4] Liu TY. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 2009,3(3):225−331. [doi: 10.1561/1500000016]

[5] Linden G, Smith B, York J. Amazon.com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, 2003, 7(1):76−80. [doi: 10.1109/MIC.2003.1167344]

[6] Breese JS, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering. In: Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers, 1998. 43−52.

[7] Herlocker JL, Konstan JA, Borchers A, Riedl J. An algorithmic framework for performing collaborative filtering. In: Proc. of the 22nd Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. ACM Press, 1999. 230−237. [doi: 10.1145/312624.312682]

[8] Sarwar B, Karypis G, Konstan J, Riedl J. Item-Based collaborative filtering recommendation algorithms. In: Proc. of the 10th Int'l Conf. on World Wide Web. ACM Press, 2001. 285−295. [doi: 10.1145/371920.372071]

[9] Bell RM, Koren Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: Proc. of the 7th IEEE Int'l Conf. on Data Mining (ICDM 2007). Omaha: IEEE, 2007. 43−52. [doi: 10.1109/ICDM.2007.90]

[10] Miyahara K, Pazzani MJ. Collaborative filtering with the simple Bayesian classifier. In: Proc. of the PRICAI 2000 Topics in Artificial Intelligence. Berlin, Heidelberg: Springer-Verlag, 2000. 679−689. [doi: 10.1007/3-540-44533-1_68]

[11] Miyahara K, Pazzani MJ. Improvement of collaborative filtering with the simple Bayesian classifier 1. 2002. [doi: 10.1007/3-540-44533-1_68]

[12] Sarwar BM, Karypis G, Konstan J, Riedl J. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In: Proc. of the 5th Int'l Conf. on Computer and Information Technology. 2002.

[13] Vucetic S, Obradovic Z. Collaborative filtering using a regression-based approach. Knowledge and Information Systems, 2005,7(1): 1−22. [doi: 10.1007/s10115-003-0123-8]

[14] Lemire D, Maclachlan A. Slope one predictors for online rating-based collaborative filtering. In: Proc. of the SDM. 2005. 1−5.

[15] Hofmann T. Latent semantic models for collaborative filtering. ACM Trans. on Information Systems, 2004,22(1):89−115. [doi: 10.1145/963770.963774]

[16] Salakhutdinov R, Mnih A, Hinton G. Restricted Boltzmann machines for collaborative filtering. In: Proc. of the 24th Int'l Conf. on Machine Learning. ACM Press, 2007. 791−798. [doi: 10.1145/1273496.1273596]

[17] Billsus D, Pazzani MJ. Learning collaborative information filters. In: Proc. of the ICML. 1998. 46−54.

[18] Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature, 1999,401(6755):788−791. [doi: 10.1038/44565]

[19] Salakhutdinov R, Mnih A. Probabilistic matrix factorization. In: Platt JC, Koller D, Singer Y, Roweis ST, eds. Proc. of the 21st Annual Conf. on Neural Information Processing Systems (NIPS 2007). Vancouver: Curran Associates, Inc., 2007.

[20] Salakhutdinov R, Mnih A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proc. of the 25th Int'l Conf. on Machine Learning (ICML 2008). New York: ACM Press, 2008. 880−887. [doi: 10.1145/1390156.1390267]

[21] Lawrence ND, Urtasun R. Non-Linear matrix factorization with Gaussian processes. In: Proc. of the 26th Annual Int'l Conf. on Machine Learning. ACM Press, 2009. 601−608. [doi: 10.1145/1553374.1553452]

[22] Rennie JDM, Srebro N. Fast maximum margin matrix factorization for collaborative prediction. In: Proc. of the 22nd Int'l Conf. on Machine Learning. ACM Press, 2005. 713−719. [doi: 10.1145/1102351.1102441]

© 中国科学院软件研究所 http://www.jos.org.cn

[23] Yu K, Zhu S, Lafferty J, Gong YH. Fast nonparametric matrix factorization for large-scale collaborative filtering. In: Proc. of the 32nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. ACM Press, 2009. 211−218. [doi: 10.1145/1571941.1571979]

[24] Koren Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Li Y, Liu B, Sarawagi S, eds. Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2008). New York: ACM Press, 2008. 426−434. [doi: 10.1145/1401890.1401944]

[25] Volkovs M, Zemel RS. Collaborative ranking with 17 parameters. In: Proc. of the Advances in Neural Information Processing Systems. 2012. 2294−2302.

[26] Koren.Y, Sill J. OrdRec: An ordinal model for predicting personalized item rating distributions. In: Mobasher B, Burke RD, Jannach D, Adomavicius G, eds. Proc. of the 5th ACM Conf. on Recommender Systems (RecSys 2011). New York: ACM Press, 2011. 117−124. [doi: 10.1145/2043932.2043956]

[27] Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. BPR: Bayesian personalized ranking from implicit feedback. In: Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence. AUAI Press, 2009. 452−461.

[28] Shi Y, Larson M, Hanjalic A. List-Wise learning to rank with matrix factorization for collaborative filtering. In: Proc. of the 4th ACM Conf. on Recommender Systems. ACM Press, 2010. 269−272. [doi: 10.1145/1864708.1864764]

[29] Liu NN, Yang Q. Eigenrank: A ranking-oriented approach to collaborative filtering. In: Proc. of the 31st Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. ACM Press, 2008. 83−90. [doi: 10.1145/1390334.1390351]

[30] Shi Y, Karatzoglou A, Baltrunas L, Larson M. TFMAP: Optimizing MAP for top-*n* context-aware recommendation. In: Hersh W, ed. Proc. of the 35th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR 2012). New York: ACM Press, 2012. 155−164. [doi: 10.1145/2348283.2348308]

[31] Shi Y, Karatzoglou A, Baltrunas L, Larson M, Oliver N, Hanjalic A. Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proc. of the 6th ACM Conf. on Recommender Systems(RecSys 2013). Dublin: ACM Press, 2012. 139−146. [doi: 10.1145/2365952.2365981]

[32] Weimer M, Karatzoglou A, Le QV, Smola A. Maximum margin matrix factorization for collaborative ranking. In: Proc. of the Advances in Neural Information Processing Systems. 2007.

[33] Weimer M, Karatzoglou A, Smola A. Improving maximum margin matrix factorization. Machine Learning, 2008,72(3):263−276. [doi: 10.1007/s10994-008-5073-7]

[34] Lee J, Bengio S, Kim S, Lebanon G, Singer Y. Local collaborative ranking. In: Proc. of the 23rd Int'l Conf. on World Wide Web (WWW 2014). Seoul, 2014. 85−96. [doi: 10.1145/2566486.2567970]

[35] Lee J, Kim S, Lebanon G, Singer Y. Local low-rank matrix approximation. In: Proc. of the 30th Int'l Conf. on Machine Learning. 2013. 82−90.

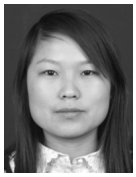
[36] Cao Z, Qin T, Liu TY, Tsai MF, Li H. Learning to rank: From pairwise approach to listwise approach. In: Proc. of the 24th Int'l Conf. on Machine Learning. ACM Press, 2007. 129−136. [doi: 10.1145/1273496.1273513]

[37] Lee J, Sun M, Lebanon G. Prea: Personalized recommendation algorithms toolkit. The Journal of Machine Learning Research, 2012, 13(1):2699−2703.

刘海洋(1987－),男,辽宁锦州人,博士生,主要研究领域为多标记分类,推荐系统.

黄丹(1990－),女,硕士生,主要研究领域为推荐系统.

王志海(1963－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为机器学习,数据挖掘.



孙艳歌(1982－),女,博士生,讲师,主要研究领域为数据挖掘,机器学习.

© 中国科学院软件研究所 http://www.jos.org.cn