软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(9):1953−1966 [doi: 10.13328/j.cnki.jos.004648]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于大规模隐式反馈的个性化推荐*

印 鉴, 王智圣, 李 琪, 苏伟杰

(中山大学 信息科学与技术学院,广东 广州 510006)

通讯作者: 印鉴, E-mail: issjyin@mail.sysu.edu.cn

摘 要: 对如何利用大规模隐式反馈数据进行个性化推荐进行了研究,提出了潜在要素模型 IFRM.该模型通过将推荐任务转化为选择行为发生概率的优化问题,克服了在隐式反馈推荐场景下只有正反馈而缺乏负反馈导致的困难.在此基础上,为了进一步提高效率和可扩展性,提出了并行化的隐式反馈推荐模型 p-IFRM.该模型通过将用户及产品随机分桶并重构优化更新序列,达到了并行优化的目的.通过概率推导,所提出的模型有坚实的理论基础.通过在 MapReduce 并行计算框架下实现 p-IFRM,并在大规模真实数据集上进行实验,可以证明所提出的模型能够有效提高推荐质量并且有良好的可扩展性.

关键词: 隐式反馈;推荐系统;大数据;MapReduce

中图法分类号: TP311

中文引用格式: 印鉴,王智圣,李琪,苏伟杰.基于大规模隐式反馈的个性化推荐.软件学报,2014,25(9):1953−1966. http://www.jos.org.cn/1000-9825/4648.htm

英文引用格式: Yin J, Wang ZS, Li Q, Su WJ. Personalized recommendation based on large-scale implicit feedback. Ruan Jian Xue Bao/Journal of Software, 2014,25(9):1953−1966 (in Chinese). http://www.jos.org.cn/1000-9825/4648.htm

Personalized Recommendation Based on Large-Scale Implicit Feedback

YIN Jian, WANG Zhi-Sheng, LI Qi, SU Wei-Jie

(School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510006, China)

Corresponding author: YIN Jian, E-mail: issjyin@mail.sysu.edu.cn

Abstract: This paper explores the area of personalized recommendation based on large-scale implicit feedback, where only positive feedback is available. To tackle the difficulty arising from lack of negative samples, a novel latent factor model IFRM is proposed, to convert the recommendation task into adoption probability optimization problem. To further improve efficiency and scalability, a parallel version of IFRM named p-IFRM is presented. By randomly partitioning users and items into buckets and thus reconstructing update sequence, IFRM can be learnt in parallel. The study theoretically derives the model from Bayesian analysis and experimentally demonstrates its effectiveness and efficiency by implementing p-IFRM under MapReduce framework and conducting comprehensive experiments on real world large datasets. The experiment results show that the model improves recommendation quality and performs well in scalability.

Key words: implicit feedback; recommendation system; big data; MapReduce

随着信息社会的迅速发展,大数据时代已经悄然而至.数据洪流将人们淹没,信息过载已成事实.为了帮助用户快速有效地获取所需,推荐系统应运而生.近年来,个性化推荐系统受到学术界和工业界的广泛关注,它通过建模用户兴趣或预测用户行为来提供信息过滤与推荐服务.

构造个性化推荐系统的关键资源是用户历史行为数据,具体可分为两类:显式用户反馈(explicit user

* 基金项目: 国家自然科学基金(61033010, 61272065, 61472453); 广东省自然科学基金(S2011020001182, S2012010009311); 广东省科技计划项目(2011B040200007, 2012A010701013)

收稿时间: 2014-04-10; 定稿时间: 2014-05-14

feedback)指用户给出的显式倾向,如电影评分、商品打分等;与之相对的是不直接表现出用户倾向的隐式用户反馈(implicit user feedback),如转发微博、浏览网站或购买商品等.收集隐式反馈数据不需要用户额外的努力,同时不易引起用户反感,因此它具有以下特点:收集成本更低、应用场景更广、数据规模更大.

大多数的推荐系统[1−6]忽视了大量的隐式反馈信息,而把注意力集中在分析挖掘显式反馈信息上,这样不仅浪费了宝贵的海量数据资源,更限制了大数据时代推荐系统的发展.因此我们认为,研究如何利用大规模隐式反馈数据进行个性化推荐,不仅体现了大数据时代的核心价值取向,还有具体的现实意义.

研究这一问题的挑战性至少包括以下 3 点:

(1) 样本不平衡.在隐式反馈推荐场景中,往往只有正反馈而没有负反馈.不像评分 1~5 可代表由“不喜欢”到“很喜欢”的倾向性程度,隐式反馈只包含“已选择”与“未选择”两类(本文将点击、交易、转发等各种交互行为称为选择),即使可以将“已选择”看成是正倾向,那么“未选择”却不能简单看成是负倾向,因为未选择的产品中混杂着用户确实不感兴趣的产品和用户未发现但感兴趣的产品(本文将电影、商品、微博等泛称为产品).这种只有正例而缺少负例的设置,给训练推荐模型带来了困难;

(2) 高维稀疏与噪声.如果使用产品作为特征表示用户的话,用户特征向量可能高达百万维.在用户-产品交互矩阵中,往往只有不到 1%的交互行为是可被观察的.相对评分而言,用户可能因为误点击产生更多噪声.高维稀疏的数据对噪声会更加敏感;

(3) 规模大.这要求推荐模型足够高效且具备良好的可扩展性,从而能够处理海量规模数据.

面对这些挑战,本文给出了解决思路.通过将推荐任务转化为用户选择行为发生概率的最大化问题,达到直接对隐式反馈数据进行建模的目的,这样既利用了“未选择”信息,又避免了在引入负例的同时引入噪声,进而提升推荐质量.借鉴降维技术,模型通过优化用户以及产品的潜在特征来建模用户与产品间的选择倾向,从而解决高维稀疏下的噪声问题.为了提高可扩展性和效率,基于分桶策略进一步提出了并行化隐式反馈推荐模型,并在 MapReduce 分布式计算框架下予以实现.

总结本文主要贡献如下:

1) 针对隐式反馈数据的特点,提出了新的潜在要素模型.该模型克服了在隐式反馈推荐场景下只有正反馈而缺乏负反馈导致的困难,通过建模用户与产品间的选择倾向来提高推荐质量;

2) 提出了并行化隐式反馈推荐模型,极大地提高了优化算法的效率和可扩展性,并给出了 MapReduce 实现方法;

3) 不仅对模型进行了理论推导,更通过大量实验证明了推荐模型及其并行化策略的有效性.

1 相关工作

1.1 基于隐式反馈的推荐

基于隐式反馈进行推荐的难点在于缺乏显式的负例,即我们明确地知道用户喜欢什么但不清楚用户不喜欢什么.Pan 等人[7]将这一类问题定义为 One-Class 协同过滤(OCCF).目前,解决这一困难的主要思路是引入负例,有 3 种策略:

(1) 利用应用环境的特殊性,人为设置规则从而指定负例,如 Jiang 等人[8]在研究微博用户的转发行为时,假设在一个有效在线会话中未被转发的微博为负例,原因是用户在这段时间很可能在线,浏览过这些微博但没有转发.但是这种策略特别依赖于特殊应用场景与人类先验知识,不具有推广性;

(2) 从未选择的产品中随机抽样作为负例[7,9,10],这类策略假设未被用户选择的产品中绝大多数是用户不会去选择的,因此通过随机抽样得到的产品可以假定为负例;

(3) 将所有未选择的产品都作为负例,但设置一个相对小的置信度(权值)[7,11,12],这种策略希望借助于这一权值来控制这些引入的不确定负例的影响.

事实上,这些方法在引入负例的同时都无可避免地引入了噪声,因为无法保证这些引入的负例中不存在潜在的正例(用户在将来会选择的产品).但对传统的推荐模型来说,引入负例又势在必行,因为只有这样才能利用

“未选择”信息.本质上,无论是随机抽样还是设置权值,都是在利用“未选择”信息与控制引入噪声中寻求平衡.引入负例的另一个缺陷是,这样做会显著增加数据规模,因此不适用于海量数据.本文与这些工作的出发点截然不同,本文试图直接建模用户选择倾向,并通过最大化可观察用户行为发生的可能性来构造和训练推荐模型,从而避免了引入负例.

1.2 协同过滤

近年来,矩阵分解已经逐渐取代传统的依赖于寻找相似用户[13]或相似产品[14]的基于邻居的协同过滤算法而成为研究热点和主流模型.这主要归因于矩阵分解有较高的预测准确率和较好的可扩展性.不像基于邻居的协同过滤算法需要计算用户-用户相似性或产品-产品相似性(这同时意味着高昂的计算开销),概率矩阵分解[1]将用户(产品)映射到低维空间并通过拟合评分信息来估计用户(产品)潜在特征向量,从而重构评分矩阵.因此,可以将矩阵分解模型简单表示为 $R\approx UV^T$,其中,R 是评分矩阵,U 是用户潜在特征矩阵,V 是产品潜在特征矩阵.这样做的好处显而易见:

(1) 利用降维技术处理高维稀疏数据以提高预测准确率;

(2) 利用稀疏的评分进行模型参数优化,避免模型计算复杂度直接随用户(产品)数增长.

但是,矩阵分解并不适合隐式反馈应用场景,这是由于缺少了关键的显式评分信息去拟合.为了利用矩阵分解模型的优越性,一些研究者将正例设为 1 而将负例设为 0.这种 0-1 矩阵分解[8,15]不可避免地需要引入负例(将全部或一些矩阵空缺项设置为 0),这样又不可避免地引入了噪声,从而影响推荐效果.

另外一种方法是将推荐任务转化为学习排序(learning to rank)问题,即通过学习构造一个排序器将用户可能感兴趣的产品排在前面.Rendle 等人[16]最早提出贝叶斯个性化排序模型,该模型通过最大化正例排在负例前面的概率,给出了协同排序的通用框架及其贝叶斯解释.Chen 等人[9]将协同排序成功应用于微博推荐任务.与以前的方法将推荐转化为 pairwise 学习排序问题不同,Shi 等人[10]进行了 listwise 学习排序方向上的进一步探索.但是这些方法始终依赖于将正例排在负例之前的初衷,因此在处理隐式反馈数据时,仍需要随机抽样一些未选择产品作为负例.

本文的方法同时借鉴了矩阵分解的降维技术以及协同排序的转化思想,但与以上相关工作存在显著不同:

(1) 不需要引入负例;

(2) 实现了并行优化框架以处理大规模数据.

因此,无论从模型角度还是从实用性角度,本文的方法更适合用于大规模隐式反馈推荐场景.

1.3 矩阵分解的并行优化算法

可以从两个不同的视角对矩阵分解的并行优化算法进行归类:其一是优化方法本身,其二是可适用于的分布式计算环境.下面分别展开讨论.

- 优化用户潜在特征矩阵 U 以及产品潜在特征矩阵 V 多采用梯度下降法,具体又可分为两种:

(1) 交替最小二乘法(alternative least sqares,简称 ALS).

通过交替地固定其中一个潜在特征矩阵(比如 U)、优化另一个潜在特征矩阵(比如 V),可以保证该算法最终收敛.设评分项个数为 $|R|$,用户数为 N,产品数为 M,潜在特征维数为 K,则 ALS 每次迭代计算复杂度较高,为 $O(|R|K^2+(M+N)K^3)$.ALS 特别适合并行化,当固定 U 后,可以并行计算 V 的每一列 $|V_j|$,反之亦然.针对 ALS 每次迭代计算复杂度较高的问题,Pilászy 等人[17]提出循环坐标下降法(cyclic coordinate descent,简称 CCD)进行了改进,该方法在固定其他变量的基础上每次只更新一个变量,加速后计算复杂度为 $O(|R|K^2)$.在此基础上,Yu 等人[18]提出了 CCD++,该方法通过改变变量更新的序列,进一步将计算复杂度降为 $O(|R|K)$.

(2) 随机梯度下降法(stochastic gradient descent,简称 SGD).

该方法每次从 R 中随机挑选一项 (i,j) 并更新相应的 U_i,V_j,更新策略为分别对 U_i,V_j 求偏导作为梯度方向,并在此梯度的负方向上前进一步,步长由学习率 α 控制.SGD 在解决矩阵分解优化问题时特别有效[3],归功于它实现简单且高效,每一次迭代计算复杂度仅为 $O(|R|K)$.但是并行化 SGD 却是一个巨大挑战,因为每一步梯度下降

依赖于上一步的更新,且并行化 SGD 会导致重写(overwriting)问题,并行更新时可能需要更新相同的 U_i 或 V_j.当 R 矩阵很稀疏时,这种重写发生的概率事实上很小,HogWild[19]正是基于这一观察通过忽略重写而被提出来.与此不同,DSGD[20,21]将矩阵 R 划分成块,通过调整更新的序列,避免了并行更新带来的重写冲突.FPSGD[22]在 DSGD 的基础上进一步优化了负载平衡与缓存利用,在实际应用中有较好的指导意义.

- 分布式计算环境可分为两种.

(1) 共享内存系统

如超级计算机、高性能服务器、GPU 等.其特点是在共享内存的基础上利用多核处理器多线程运算.

(2) 分布式系统

网格、集群可归为此类,其特点是内存不共享,利用多台计算机组成的网络实现并行计算.在这类系统中,目前最为流行的是 MapReduce 计算框架[23],它将海量数据的处理任务抽象为 Map 过程和 Reduce 过程,其中,Map 的作用是将原始海量数据映射为不同的数据子集,Reduce 的作用是对 Map 处理后的结果进行规约.对于矩阵分解并行优化算法来说,也可根据其适用于的环境分为两类:其一是只适用于共享内存系统的,如 Zhuang 等人[22]依赖于共享内存的高效访问,但同时也存在规模限制,如要求共享内存足够完全装载特征矩阵 U,V;另一类是通用方法,并没有对分布式计算环境有特殊要求,如 Schelter 等人[24]、Gemulla 等人[20]都基于 MapReduce 框架实现了其并行优化算法.

前面提到的并行优化算法都应用在显示反馈推荐场景,并行的目的是为了尽可能快地逼近传统矩阵分解的优化目标(最小平方误差和).本文提出的并行化优化方法根据隐式反馈推荐模型的特点并考虑到优化算法的通用性,选择 SGD 作为基本优化算法,并基于 MapReduce 框架予以实现.

2 问题定义和基于隐式反馈的推荐模型

2.1 问题定义

我们首先形式化地定义基于隐式反馈的推荐问题.推荐模型可以利用的信息是用户-产品历史选择集合,如图 1 左侧的矩阵所示,其中,灰色代表观察到的用户选择行为,白色代表用户尚未选择该产品.当给定任意用户 i 以及待推荐列表 L(由待推荐候选产品组成),推荐模型可以生成 L 的排序 L^*,L^*应尽可能地将用户将要选择的产品排在其他产品前面.而需要解决的问题是,如何根据历史选择集合构造出这样一个推荐模型.

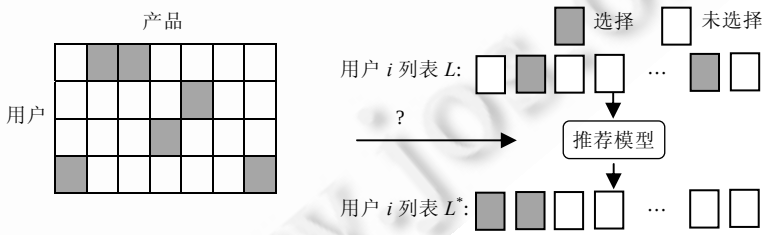

Fig.1 Problem definition of recommendation based on implicit feedback

图 1 基于隐式反馈推荐的问题定义

2.2 基于隐式反馈的推荐模型IFRM

受矩阵分解模型的启发,我们假设任意用户 i 与任意产品 j 可以在潜在 K 维特征空间表示为一组向量:$U_i=(U_{i1},U_{i2},\ldots,U_{iK})$,$V_j=(V_{j1},V_{j2},\ldots,V_{jK})$,并且用户的选择行为由用户的潜在特征与产品的潜在特征共同决定,令 A_{ij} 表示用户 i 对产品 j 的选择倾向程度,则

$$A_{ij}=U_i\cdot V_j=\sum_{k=1}^{K}U_{ik}V_{jk} \tag{1}$$

事实上,这种绝对的选择倾向程度本身不足以反映用户的选择行为,应该在比较下才有意义.比如用户对某

产品的选择倾向程度为 10,但这不意味着用户会选择该产品或者不会选择该产品.当用户对大多数产品的选择倾向都很高时(平均选择倾向程度为 20),那么用户很可能不会选择该产品;反之,当用户对大多数产品的选择倾向都很低时(平均选择倾向程度为 5),那么用户很可能会选择该产品.综上,我们认为:用户 i 之所以选择产品 j,是因为对用户 i 来说,产品 j 比一般产品更具吸引力.为了体现这种相对性,我们定义:

$$\Delta_{ij}=\frac{A_{ij}}{\overline{A_i}}=\frac{A_{ij}}{\frac{1}{M}\sum_{h=1}^{M}A_{ih}}.$$

其中,$\overline{A_i}$ 表示用户 i 对全体 M 个产品的平均选择倾向.用户 i 选择产品 j 的概率与 Δ_{ij} 相关:Δ_{ij} 越大,表示用户 i 越可能选择产品 j;Δ_{ij} 越接近于 0,表示用户 i 越不可能选择产品 j.进一步地,我们利用 sigmoid 函数 $\varphi(x)=\frac{x}{1+x}$ 将这种可能性归一化到(0,1)区间,作为用户 i 选择产品 j 的概率 $P_{ij}=\varphi(\Delta_{ij})=\frac{\Delta_{ij}}{1+\Delta_{ij}}$.

设集合 $O=\{\langle i,j\rangle|$用户 i 选择产品 $j\}$是观察到的选择行为的集合,假设每一次的选择行为是独立的,那么集合 O 的似然概率为 $P(O|U,V)=\prod_{\langle i,j\rangle\in O}P_{ij}=\prod_{\langle i,j\rangle\in O}\frac{1}{1+\Delta_{ij}^{-1}}$.

训练推荐模型的本质是:给定观察集 O(也即是训练集),最大化潜在特征 U,V 的后验概率 $P(U,V|O)$.由贝叶斯定理:$P(U,V|O)=\frac{P(O|U,V)P(U)P(V)}{P(O)}\propto P(O|U,V)P(U)P(V)$,当假设 U,V 服从均值为 0、方差为 σ^2 的高斯先验时,$P(U,V|O)\propto\prod_{\langle i,j\rangle\in O}\frac{1}{1+\Delta_{i,j}^{-1}}\prod_{i,k}N(U_{ik}|0,\sigma^2)\prod_{j,k}N(V_{jk}|0,\sigma^2)$.

取对数:$\ln P(U,V|O)\propto-\sum_{\langle i,j\rangle\in O}\ln(1+\Delta_{ij}^{-1})-\frac{1}{2\sigma^2}\|U\|_F^2-\frac{1}{2\sigma^2}\|V\|_F^2$,其中,$\|*\|_F^2$ 表示 2 范数.于是,最大化后验概率 $P(U,V|O)$等价于最小化以下优化目标:

$$\arg\min_{U,V}L:=\sum_{\langle i,j\rangle\in O}\ln(1+\Delta_{ij}^{-1})+\lambda(\|U\|_F^2+\|V\|_F^2) \tag{2}$$

其中,$\lambda=\frac{1}{2\sigma^2}$ 是调节参数,用于调节控制潜在特征矩阵 U,V 的复杂度,以防止过拟合.

至此,IFRM 将基于隐式反馈的推荐问题转化为一个最优化问题,即根据可观察选择行为集合 O 求得潜在特征矩阵 U,V.在预测时,对任意给定的用户 i 以及候选产品集$\{j_1,j_2,\ldots,j_p\}$,根据由公式(1)计算出的 A_{ij} 由高到低对候选产品进行排序,即可产生推荐列表.

2.3 IFRM的优势分析

IFRM 借鉴潜在因子模型处理高维稀疏数据的思路,利用降维技术在低维空间对用户以及产品建模,提高了模型抗噪能力.IFRM 继承了矩阵分解的良好可扩展性,模型复杂程度仅随可观察的选择行为数$|O|$线性增长.这与传统的基于用户或基于产品的协同过滤方法有显著不同,当用户或产品规模庞大时,计算用户-用户相似性或产品-产品相似性是非常耗时的;

与传统的基于显示反馈推荐方法不同,IFRM 并不去拟合具体评分值,而是去最大化可观察用户行为出现的概率.通过建模用户选择倾向而不是评分行为,IFRM 天然地适用于没有评分而只有选择交互行为的隐式反馈推荐场景;

基于隐式反馈的矩阵分解(iMF)[11]与贝叶斯个性化排序(BPR)[16]等在隐式反馈场景下进行推荐的方法有个共同潜在假设,即:用户表现出来的反馈就是正样本,未表现出来的就是负样本.基于此,iMF 将正样本设置为 1 而将负样本设置为 0,而 BPR 则最大化正样本排在随机抽样得到的“负”样本前面的概率.然而,这些负样本却有可能是潜在的正样本,即在将来会发生的选择行为.这样,难以避免会引入噪声(假负).与它们不同,IFRM 并不进行这样的假设,通过关注选择行为本身,并最大化选择行为发生的概率,避免了引入噪声,进而提高推荐质量.

2.4 优化算法

通过第 2.2 节的建模与推导,基于隐式反馈的推荐问题已经被转化为优化问题.由于优化目标(2)不是传统矩阵分解最小平方误差和的形式,不能使用交替最小二乘法(ALS)进行优化.本文选择随机梯度下降法(SGD)进行优化.对观察到的选择行为$\langle i,j\rangle$,分别对 U_i,V_j 求偏导作为梯度:

$$\frac{\partial L}{\partial U_i}=f(i,j)\bar{V}-g(i,j)V_j+2\lambda U_i \tag{3}$$

$$\frac{\partial L}{\partial V_j}=-g(i,j)U_i+2\lambda V_j \tag{4}$$

其中,$f(i,j)=\frac{1}{U_iV_j+U_i\bar{V}}$,$g(i,j)=\frac{U_i\bar{V}}{U_iV_j}f(i,j)=\frac{U_i\bar{V}}{U_iV_j(U_iV_j+U_i\bar{V})}$,$U_i$ 表示用户 i 的特征向量,V_j 表示产品 j 的特征向量,$\bar{V}=\frac{1}{M}\sum_{j=1}^{M}V_j$ 表示全部产品的平均特征向量.

使用随机梯度下降法求解 IFRM 的基本算法如下:

算法 1. SGD 求解 IFRM.

输入:可观察选择集合 O,学习率α,最大迭代次数 T;

输出:潜在特征矩阵 U,V.

1. 随机初始化 U,V
2. for $iter$=1,2,…,T do:
3. for each $\langle i,j\rangle\in O$:
4. 根据公式(3)所求梯度更新 $U_i=U_i-\alpha\frac{\partial L}{\partial U_i}$
5. 根据公式(4)所求梯度更新 $V_j=V_j-\alpha\frac{\partial L}{\partial V_j}$
6. end for
7. end for

3 并行化的基于隐式反馈的推荐模型及其 MapReduce 实现

在基于隐式反馈的推荐场景中,数据规模往往特别巨大,这要求推荐模型足够高效且有良好的可扩展性.本节将进一步介绍并行化的隐式反馈推荐模型(p-IFRM),并在 MapReduce 分布式计算框架下对其进行了实现.

3.1 并行化隐式反馈推荐模型p-IFRM

本质上,并行化算法 1 的困难是由 SGD 必须序列化逐步迭代引起的,具体来说,SGD 每一次求解梯度依赖于上一步的更新.一般的,直接将 SGD 并行化会导致两个问题:

(1) 延迟更新,即本次的更新将会在 q 步之后生效,q 为并行线程数或计算节点数.这是由于为了并行,我们不得不根据 q 步之前的状态来计算梯度,这就相当于将每次更新推迟在了 q 步之后.Langford, Agarwal 等人[25,26]证明了这种延迟更新不会影响算法最终收敛以及优化效果;

(2) 重写问题,即有可能正在并行运算的节点同时要求更新同一变量,于是产生写冲突.

如果将更新变量划分为不相交的子集,那么并行化 SGD 的困难就可迎刃而解.正是基于这一思路,本文提出利用分桶策略并行化 IFRM.在进行每一次迭代之前,首先将用户以及产品随机分桶,然后根据分桶结果产生可并行优化的不冲突子集,在每个子集上独立地执行 SGD 算法.这一方法在本质上是通过改变和限制变量的更新优化序列来避免重写冲突,实现并行.

图 2 使用了简单示例来说明 p-IFRM 的框架.假设原可观察选择集合 O 由 6 个用户与 9 个产品之间的选择交互行为构成,其中,灰色代表选择,白色代表未选择.首先将用户与产品随机分为 Q 桶,一般地,Q 为计算节点数.

这里,假设有 3 个计算节点,于是,6 个用户分别被分到桶 1,2,3;9 个产品分别被分到桶 *A*,*B*,*C*.通过合理组合用户桶与产品桶,我们可得到不相交组合(1*A*,2*B*,3*C*),(1*B*,2*C*,3*A*)以及(1*C*,2*A*,3*B*).注意,这里的不相交指的是在同一个组合中,桶号 1,2,3,*A*,*B*,*C* 分别仅出现 1 次.这种不相交组合避免了并行优化过程中冲突的产生.于是,原可观察选择集合 *O* 可被划分为 9 个子集,并按生成的组合分配给不同的计算节点.每个计算节点可根据算法 1 独立地进行优化.

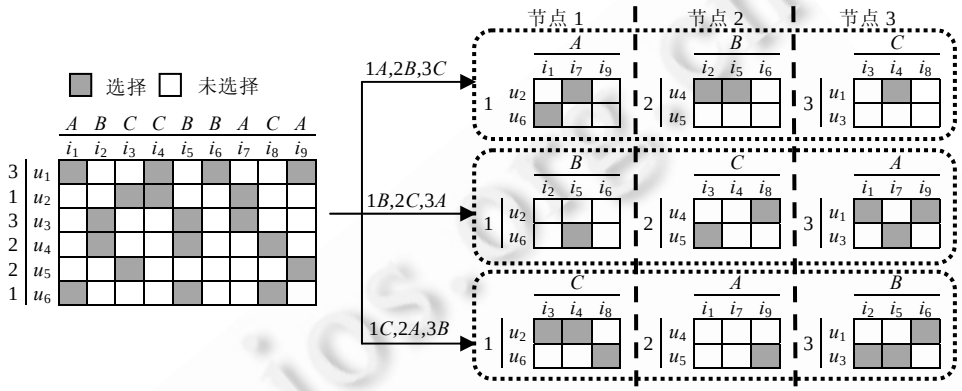

Fig.2 An illustration of p-IFRM framework

图 2 p-IFRM 框架示例

p-IFRM 的基本算法由算法 2 给出.需要说明的是,为了保证算法稳定性,每一次外层迭代都将用户与产品重新分桶,内层循环保证了所有观察数据都能够被利用.在确定好训练子集后,每个计算节点只需要根据分配给它的训练子集装载与本次训练相关的部分 *U*,*V* 即可,这相当于同时对潜在特征矩阵 *U*,*V* 进行了拆分.利用共享内存来存放所有变量的并行优化算法往往要求内存可以容纳全部 *U*,*V*,然而 p-IFRM 并不存在这一限制.这极大地提高了 p-IFRM 的可扩展性.

算法 2. p-IFRM.

输入:可观察选择集合 *O*,学习率α,最大迭代次数 *T*,计算节点数 *Q*;

输出:潜在特征矩阵 *U*,*V*.

1. 随机初始化 *U*,*V*
2. for *iter*=1,2,…,*T* do:
3. 将用户以及产品随机分桶
4. for *q*=1,2,…,*Q* do:
5. 生成本次分桶组合并分配相应 *Q* 个训练子集
6. 在 *Q* 个计算节点分别装载训练子集以及所需的部分 *U*,*V*
7. 使用 SGD 算法在各计算节点并行优化
8 end for
9. end for

3.2 p-IFRM的MapReduce实现

MapReduce 是一个高度抽象的通用并行化计算框架,它通过将运算抽象为 Map 过程与 Reduce 过程来处理大规模问题.由于 Map 过程与 Reduce 过程本质上都是将原始键值对映射为新的键值对,因此实现 MapReduce 的关键在于定义合适的键值对及其映射方式.

利用 Map 将训练数据 *O* 映射划分到各计算节点,然后利用 Reduce 对划分后不冲突的 *Q* 个子集进行训练.注意到:在此过程中,Map 过程(数据划分过程)与 Reduce 过程(优化过程)都是并行执行的,即并行随机地读取训

练集合 O 并进行划分,然后并行地对 U,V 进行优化,最后并行地输出更新后的 U,V 以完成一次迭代过程.

我们进一步给出 Map 与 Reduce 实现方法:

算法 3. Map 方法.

输入:⟨用户 ID,产品 ID⟩;

输出:⟨桶 ID,(用户 ID,产品 ID)对⟩.

装载:本次分桶结果以及分桶组合

1. 根据分桶结果将(用户 ID,产品 ID)映射为桶 ID
2. if 桶 ID 在分桶组合中:
3. 输出⟨桶 ID,(用户 ID,产品 ID)⟩
4. else:
5. 丢弃该样本
6. end if

算法 4. Reduce 方法.

输入:⟨桶 ID,(用户 ID,产品 ID)对列表 L⟩;

输出:⟨用户 ID,U_i⟩或⟨产品 ID,V_j⟩.

装载:相关 U,V

1. for each ⟨i,j⟩ in L:
2. 根据公式(3)所求梯度更新 U_i
3. 根据公式(4)所求梯度更新 V_j
4. end for
5. 输出更新后的 U,V

算法 3 中的桶 ID 可以是形如"1*A*","2*C*"的关键字,在运行 Map 方法之前需要装载本次的分桶方法(可以是一个字典,将用户或产品映射为桶号)以及分桶组合(形如["1*A*","2*B*","3*C*"]的列表).以 Apache Hadoop[27]为例,可以通过重写 setup 函数以实现装载功能.算法 4 同样需要在运行 Reduce 之前装载最新的与本次优化相关的 U,V,算法最终输出的是更新后的用户或产品的特征向量.Reduce 的输出可以作为下一次迭代优化需要装载的输入,也可以作为产生推荐结果的模型变量.

4 实 验

本节首先介绍实验所用数据集以及实验设置,然后说明评价标准以及对比算法,最后给出 IFRM 与其他方法的对比实验结果及其 MapReduce 实现的并行化效果,并对实验结果进行了相应分析.

4.1 数据集描述及实验设置

本文采用两个数据集.详细统计信息见表 1.

Table 1 Data set description

表 1 数据集描述

		用户	产品	选择行为
豆瓣	训练集	33 213	84 633	5 095 277
	测试集	8 156	23 982	151 972
Lastfm	训练集	977	169 325	12 123 422
	测试集	799	94 347	233 657

为了拟合真实推荐场景,我们按照时间将数据切分为训练集与测试集.对测试集中的每个用户,随机抽样了 1 000 个待推荐产品作为候选集,其中包括用户在测试集中选择的产品以及用户在训练集与测试集中从未选择过的产品.各推荐模型的目标是根据切分时间点之前的隐式反馈信息推测未来的用户选择行为并生成推荐列

表(如图1右侧所示),即将候选产品集根据被用户选择的可能性由高到低进行排序.好的推荐算法应该能够将用户确实选择的产品尽可能地排在前面.由于冷启动问题不在本文讨论范围之内,且对比算法无法解决冷启动问题,为保障实验的公平性,我们过滤掉测试集中新出现的用户和产品.

4.2 评价标准

实验使用平均正确率均值(mean average precision,简称 MAP)以及平均百分比排序(mean percentage ranking,简称 MPR)作为评价指标.

MAP 是信息检索领域常用评价指标,它用来评价算法的整体预测准确率.MAP 是对所有测试用户的平均正确率(average precision,简称 AP)的再一次平均.给定一个用户 i 以及他的长度为 M 的已排序推荐列表 $\langle j_1,j_2,\ldots,j_M\rangle$,假设用户 i 选择了其中 N 个,则可以计算出平均正确率:

$$AP_i=\frac{\sum_{k=1}^{M} precision(k)\times ref(k)}{N},$$

其中,$precision(k)$是 Top-k 的准确率,如果 j_k 命中,则 $ref(k)$=1;否则,$ref(k)$=0.MAP 这一评价指标特别看重推荐列表的前几位是否命中,MAP 越高,表示算法的推荐准确率越高.

MPR 常被用来衡量用户对推荐列表的满意程度,是一种面向召回率的评价指标.我们把为用户 i 提供的已排序推荐列表中产品 j 的排序位置记为 $rank_{i,j}$,如果 j 排在推荐列表首位,则 $rank_{i,j}$=0%;如果排在末位,则 $rank_{i,j}$=100%.通过平均所有被用户选择的产品的排序位置,可得:

$$MPR=\frac{1}{|Users|}\sum_i\frac{ind(i,j)\times rank_{i,j}}{\sum_j ind(i,j)},$$

其中,$|User|$表示测试集中用户的总数;$ind(i,j)$是指示函数,如果用户 i 选择了产品 j 则为 1,否则为 0.这一评价指标看重预测平均召回率.注意到,完全随机的推荐列表的 MPR 值应该接近 50%.MPR 越低,表示算法在预测召回率上的表现越好.

4.3 对比算法

根据隐式反馈推荐应用场景的特点,本文选择 4 种算法作为对比算法:

- 基于用户的协同过滤(user-based CF)[13]:该算法首先根据历史选择信息为每个用户计算出相似用户集,然后根据这些相似用户的共同投票进行推荐;
- 基于产品的协同过滤(item-based CF)[14]:该算法首先根据历史选择信息为每个产品计算出相似产品集,然后通过平均用户对这些相似产品的选择情况来估计待推荐产品有多大可能被用户选择;
- 基于隐式反馈的矩阵分解(iMF)[11]:通过将可观察的选择设为 1,将未选择设为 0,并增加置信度参数(权重),该方法将传统 0-1 矩阵分解成功应用于隐式反馈推荐场景;
- 贝叶斯个性化排序(BPR)[16]:通过将推荐问题转化为 pairwise 排序问题,并从不可观察数据中为每个正例随机抽样得到一个负例形成对(pair),该方法也可用于隐式反馈推荐场景.

4.4 实验结果与分析

实验环境配置说明:本文提出的模型在 Hadoop(版本 1.2.1)集群上进行实验,集群由 8 台 PC 机组成,每台 PC 机配备酷睿 i3 双核处理器,2G 内存.对比算法采用单机环境,在其中 1 台 PC 机上进行实验.

实验 1.推荐效果.

该实验通过将 IFRM 与 4 种算法进行对比证明 IFRM 的有效性.为了获得更可靠的实验结果,我们将实验重复 10 次,每次重新构造测试集(为每个目标用户重新随机生成 1 000 个候选产品).由表 2 可以看出,IFRM 在 MAP 评价指标上有较大优势,这说明 IFRM 可以显著提高 Top$N$ 推荐的命中率.在 MPR 评价指标上,IFRM 在 lastfm 数据集中表现出了一致的优越性,在豆瓣数据集中与传统协同过滤方法以及 BPR 相比,取得了不错的效果(分别降低了 39.0%,33.4%,11.4%),但是与 iMF 相比并没有明显优势,原因在于 iMF 专注于优化误差平方和,该优化目

标与 MPR 注重召回率更为一致.尽管如此,IFRM 在 MAP 评价指标上明显地优于 iMF,这是由于 iMF 过分关注整体误差而忽视了排在前几位的命中率.在实际应用中,Top$N$ 推荐命中率尤其重要,因为用户不可能耐心浏览完所有推荐产品.与 iMF 以及 BPR 相比,IFRM 可以产生更高质量的推荐,因为它直接对用户选择行为进行建模,从而避免了引入噪声.

Table 2 Performance comparison of IFRM and baselines

表 2 IFRM 与对比算法推荐效果比较

数据集	对比算法	MAP (%)	IFRM 提升 MAP (%)	MPR (%)	IFRM 降低 MPR (%)
豆瓣	User-Based CF	0.131±0.011	10.7	19.69±2.17	39.0
	Item-Based CF	0.135±0.015	7.4	18.04±1.75	33.4
	iMF	0.121±0.012	19.8	12.02±1.10	0.0
	BPR	0.139±0.011	4.3	13.57±1.55	11.4
	IFRM	0.145±0.010	–	12.02±0.97	–
Lastfm	User-Based CF	0.379±0.021	11.1	34.65±2.33	6.0
	Item-Based CF	0.385±0.018	9.4	33.82±1.29	3.7
	iMF	0.367±0.013	14.7	33.48±2.02	2.7
	BPR	0.412±0.018	2.2	33.51±1.73	2.8
	IFRM	0.421±0.011	–	32.56±1.24	–

实验 2.并行优化效果以及加速效果.

首先,通过改变计算节点数 Q(也即分桶数)比较 p-IFRM 在不同的计算资源下的表现.图 3 与图 4 展示了 p-IFRM 在豆瓣数据集上的优化效果:当 Q=1 时,p-IFRM 退化为 IFRM,即在单个节点上非并行运算.可以看到:IFRM 的优化曲线比较平滑,而 p-IFRM 在最初的几次迭代时存在一些颠簸,这是由于随机分桶造成的;但在一定的迭代次数之后(大于 10 次),各模型的预测能力趋于一致,这说明只要经过多次迭代,IFRM 与 p-IFRM 都将收敛,且不会有明显差异.这证明了基于分桶策略的 p-IFRM 是有效的.同时可以看到:最开始的几次迭代,IFRM 的收敛速度要明显快于 p-IFRM,这是因为每个计算节点都根据经过划分后的小数据子集来进行优化,这种划分难免会造成部分信息的丢失;但是通过多次重新分桶,这种信息丢失会被逐渐弥补,这也是最终优化效果趋于一致的原因之一.值得注意的是:最终收敛时,p-IFRM 能够取得更低的 MPR 值;并且随着分桶数的增加,模型在 MPR 上的表现越来越好.这是由于在较小规模子集上更容易挖掘出用户的特殊喜好或产品的特殊特性,从而提高预测召回率.

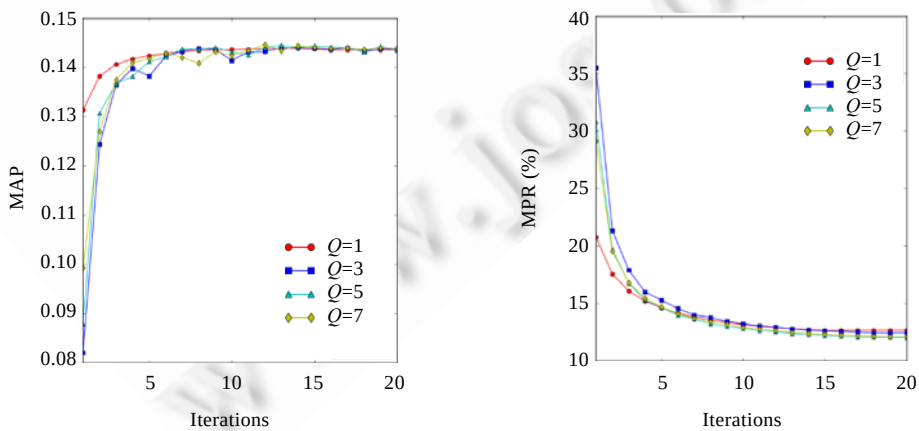

Fig.3 Effect of different number of computation nodes Q (number of buckets) on optimization performance

图 3 计算节点数 Q(分桶数)对优化效果的影响

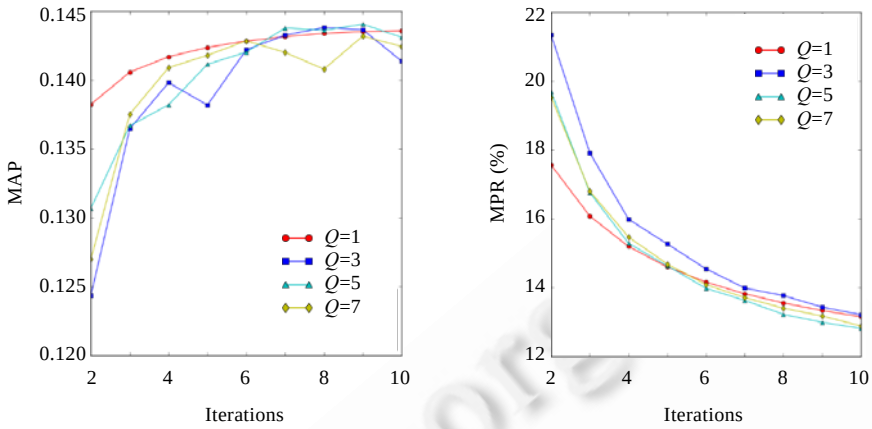

Fig.4 Effect of different number of computation nodes Q (number of buckets) on optimization performance (zoom in iteration 2~10)

图 4 计算节点数 Q(分桶数)对优化效果的影响(放大观察 2~10 次迭代)

图 5 展示了增加计算资源对并行化加速的影响,可以看出:增加计算节点将显著减少每次迭代所需时间,这说明 p-IFRM 能够通过增加计算节点处理更大规模数据.值得注意的是:随着增加计算资源,加速效果逐渐放缓.这是因为在 Hadoop 集群中,通信和 I/O 等时间消耗不可避免,增加计算资源会同时增加这些消耗.

通过在豆瓣数据集上对用户和产品进行抽样(抽样率为 30%~100%),我们进一步测试了各算法在不同规模数据集上的运行效率.从图 6 可以看出:iMF 效率最低,主要是因为 iMF 将所有不可观察的交互行为都当成负例,因此导致训练数据规模庞大.随着用户数以及产品数的增多,传统的基于邻居的协同过滤算法的运行时间呈指数增长,可扩展性较差.这是由于这类算法需要计算用户-用户相似性或产品-产品相似性,所以算法效率对用户和产品数特别敏感.BPR 以及 IFRM 模型都只针对可观察用户-产品交互进行模型优化,因此具有更好的可扩展性.其中,p-IFRM 利用并行化技术进一步提高了效率,随着数据规模的增大,算法运行时间并不会显著增加.

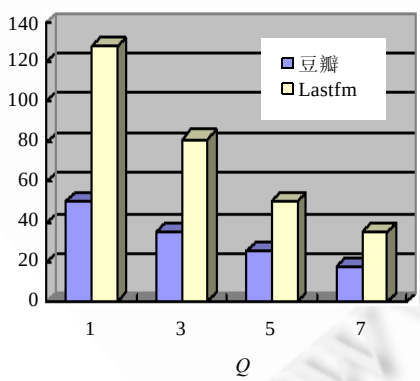

Fig.5 Effect of different number of computation nodes Q (number of buckets) on speeding up

图 5 计算节点数 Q(分桶数)对加速效果的影响(纵坐标单位为秒)

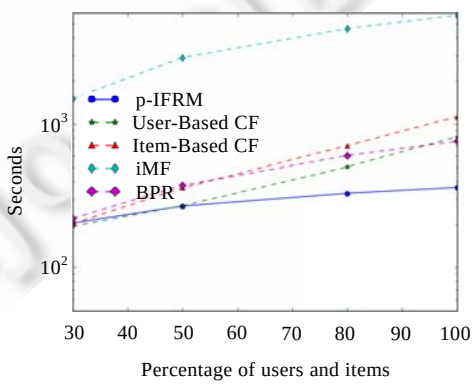

Fig.6 Total running time of p-IFRM and baselines (Q=7)

图 6 p-IFRM 与对比算法总体运行时间比较(Q=7)

实验 3.参数影响.

IFRM 中需要设定 3 个参数,分别是潜在特征维数 K、正则化参数λ以及学习率α.我们进一步研究了不同的

参数设置对实验结果产生的影响.以下实验均在豆瓣数据集上进行.

由表 3 可以看出,推荐效果对特征维数不敏感.特征维数 K 的建议取值为 10~20 之间.优化过程耗时会随着维数的增加显著增加,这是因为每一次更新都需要计算 K 维向量的点积,而 I/O 大小也与维数 K 直接相关.当 K 取 50 时,模型过分拟合训练数据,反而对结果产生不良影响.

Table 3 Effect of different latent feature dimension K (λ=10^{-5}, Q=5, α=0.01)

表 3 潜在特征维数 K 的影响(λ=10^{-5},Q=5,α=0.01)

	K=5	K=10	K=20	K=50
MAP	0.141 35	0.143 49	0.144 01	0.143 31
MPR (%)	12.814	12.019	12.011	12.036
平均迭代一次耗时(s)	19.18	25.1	44.65	108.57

由表 4 可以看出:正则化参数λ应该设置一个较小的值,这说明引入正则化项是有效的,但作用不明显.通过观察发现:当将λ设置为 0 时,优化后的潜在特征矩阵中的数值依然较小(大部分处于 0.5~1.0 之间),这说明不像传统的矩阵分解,IFRM 在避免潜在特征矩阵过于复杂防止过拟合方面有天然优势,这归因于模型优化被选产品与一般产品之间的相对比例,而不是去拟合一个绝对数值.

Table 4 Effect of different regularization parameter λ (K=10, Q=5, α=0.01)

表 4 正则化参数λ的影响(K=10,Q=5,α=0.01)

	λ=10^{-1}	λ=10^{-3}	λ=10^{-5}	λ=0
MAP	0.132 49	0.135 57	0.143 49	0.142 93
MPR (%)	13.591	13.014	12.019	12.572

对于随机梯度下降法来说,学习率α是一个重要参数,将显著影响收敛速度以及优化效果.由图 7 可以看出,将α设置为 0.01 左右较为合适.过大的学习率会导致模型无法收敛,当将α设置为 0.03 时,经过 3 次迭代推荐效果就已经达到顶峰无法继续提升,并且在经过第 8 次迭代之后,推荐效果有明显的下降趋势;过小的学习率会导致收敛速度过慢,当将α设置为 0.001 时,虽然最终可以收敛并达到理想的推荐效果,但需要经过 40 次迭代(图 7 中只画出了前 10 次).一个常用的启发式的学习率设定方法为:动态调整学习率,随迭代次数逐渐减小.更详细的学习率设定方法超出了本文的研究范围,在此不再赘述.

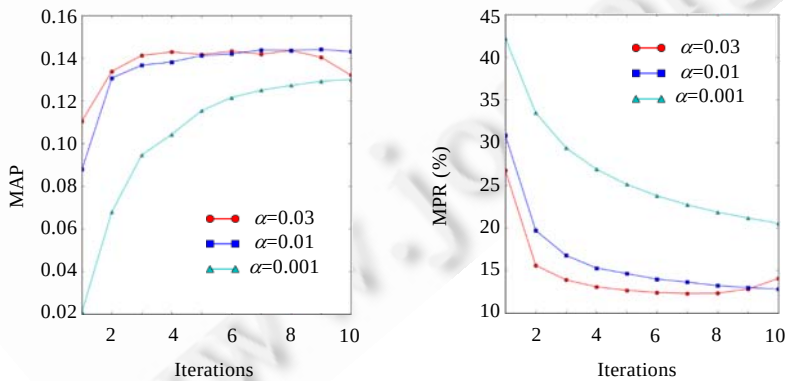

Fig.7 Effect of different learning rate α

图 7 学习率α的影响

5 总 结

针对如何在大数据时代提高个性化推荐方法的精度与效率的问题,本文研究了基于大规模隐式用户反馈数据的高效推荐算法设计,提出了一种新颖的基于隐式反馈的推荐模型 IFRM,它通过对可观察用户选择行为数据集(隐式用户反馈数据集)的后验估计,将推荐问题转化为优化问题.为了提高 IFRM 的可扩展性,进一步提

出了一个基于分桶策略的并行优化模型 p-IFRM,并提供了 MapReduce 实现.最后,实验结果验证了所提算法模型及其并行策略的有效性.

下一步,我们将搭建更大规模的集群平台,并尝试对 TB 以上规模的隐式反馈数据进行建模.我们还将尝试整合隐式反馈数据与社交数据,将 IFRM 应用于社交网络推荐场景.此外,虽然冷启动问题并不在本文研究范围之内,但是与传统方法相比,本文所提模型 IFRM 在解决冷启动问题时更具潜力,可以通过引入一些潜在的用户选择行为来克服冷启动和数据稀疏性.最后,本文设定的应用场景为只有隐式反馈而没有显式反馈,虽然这种应用场景非常普遍,但在现实生活中,隐式反馈也可能与显式反馈并存,如何整合这两种数据资源以提高推荐质量,有待进一步研究.

致谢 本文部分工作是在作者访问中国人民大学的萨师煊大数据研究中心时完成的.该中心获国家高等学校学科创新引智计划(111 计划)等资助.在此,我们向对本文的工作给予支持和建议的同行表示感谢.特别感谢评审老师提出的宝贵意见.

References:

[1] Salakhutdinov R, Mnih A. Probabilistic matrix factorization. In: Proc. of the Advances in Neural Information Processing Systems. 2007. 1257−1264.

[2] Zhou Y, Wilkinson D, Schreiber R, Pan R. Large-Scale parallel collaborative filtering for the netflix prize. In: Proc. of the Algorithmic Aspects in Information and Management. Berlin, Heidelberg: Springer-Verlag, 2008. 337−348. [doi: 10.1007/978-3-540-68880-8_32]

[3] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer, 2009,42(8):30−37. [doi: 10.1109/MC.2009.263]

[4] Liu X, Aberer K. SoCo: A social network aided context-aware recommender system. In: Proc. of the 22nd Int'l Conf. on World Wide Web. 2013. 781−802.

[5] Chen J, Yin J. A collaborative filtering recommendation algorithm based on influence sets. Ruan Jian Xue Bao/Journal of Software, 2007,18(7):1685−1694 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1685.htm [doi: 10.1360/jos181685]

[6] Sun GF, Wu L, Liu Q, Zhu C, Chen EH. Recommendations based on collaborative filtering by exploiting sequential behaviors. Ruan Jian Xue Bao/Journal of Software, 2013,24(11):2721−2733 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4478.htm [doi: 10.3724/SP.J.1001.2013.04478]

[7] Pan R, Zhou Y, Cao B, Liu NN, Lukose R. One-Class collaborative filtering. In: Proc. of the 8th IEEE Int'l Conf. on Data Mining (ICDM 2008). IEEE, 2008. 502−511. [doi: 10.1109/ICDM.2008.16]

[8] Jiang M, Cui P, Liu R, Yang Q, Wang F, Zhu W, Yang S. Social contextual recommendation. In: Proc. of the 21st ACM Int'l Conf. on Information and Knowledge Management. ACM Press, 2012. 45−54. [doi: 10.1145/2396761.2396771]

[9] Chen K, Chen T, Zheng G, Jin O, Yao EP, Yu Y. Collaborative personalized tweet recommendation. In: Proc. of the 35th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. ACM Press, 2012. 661−670. [doi: 10.1145/2348283.2348372]

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

[11] Hu Y, Koren Y, Volinsky C. Collaborative filtering for implicit feedback datasets. In: Proc. of the 8th IEEE Int'l Conf. on Data Mining (ICDM 2008). IEEE, 2008. 263−272. [doi: 10.1109/ICDM.2008.22]

[12] Steck H. Training and testing of recommender systems on data missing not at random. In: Proc. of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2010. 713−722. [doi: 10.1145/1835804.1835895]

[13] 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]

[14] 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]

[15] Kurucz M, Benczur AA, Kiss T, Nagy I, Szabo A, Torma B. Who rated what: A combination of SVD, correlation and frequent sequence mining. In: Proc. of the KDD Cup and Workshop, Vol.23. 2007. 720−727.

[16] 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.

[17] Pilászy I, Zibriczky D, Tikk D. Fast als-based matrix factorization for explicit and implicit feedback datasets. In: Proc. of the 4th ACM Conf. on Recommender Systems. ACM Press, 2010. 71−78. [doi: 10.1145/1864708.1864726]

[18] Yu HF, Hsieh CJ, Si S, Dhillon IS. Parallel matrix factorization for recommender systems. Knowledge and Information Systems, 2013. 1−27.

[19] Niu F, Recht B, Ré C, Wright SJ. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. arXiv preprint arXiv: 1106.5730, 2011.

[20] Gemulla R, Haas PJ, Nijkamp E, Sismanis Y. Large-Scale matrix factorization with distributed stochastic gradient descent. In: Proc. of the 17th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2011. 69−77. [doi: 10.1145/2020408.2020426]

[21] Recht B, Ré C. Parallel stochastic gradient algorithms for large-scale matrix completion. Mathematical Programming Computation, 2011. 1−26.

[22] Zhuang Y, Chin WS, Juan YC, Lin CJ. A fast parallel SGD for matrix factorization in shared memory systems. In: Proc. of the 7th ACM Conf. on Recommender Systems. ACM Press, 2013. 249−256. [doi: 10.1145/2507157.2507164]

[23] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008,51(1): 107−113. [doi: 10.1145/1327452.1327492]

[24] Schelter S, Boden C, Schenck M, Alexandrov A, Markl V. Distributed matrix factorization with mapreduce using a series of broadcast-joins. In: Proc. of the 7th ACM Conf. on Recommender Systems. ACM Press, 2013. 281−284. [doi: 10.1145/2507157.2507195]

[25] Langford J, Smola A, Zinkevich M. Slow learners are fast. arXiv preprint arXiv: 0911.0491, 2009.

[26] Agarwal A, Duchi JC. Distributed delayed stochastic optimization. In: Proc. of the IEEE 51st Annual Conf. on Decision and Control (CDC 2012). IEEE, 2012. 5451−5452.

[27] Apache hadoop. http://hadoop.apache.org

附中文参考文献:

[5] 陈健,印鉴.基于影响集的协作过滤推荐算法.软件学报,2007,18(7):1685−1694. http://www.jos.org.cn/1000-9825/18/1685.htm [doi: 10.1360/jos181685]

[6] 孙光福,吴乐,刘淇,朱琛,陈恩红.基于时序行为的协同过滤推荐算法.软件学报,2013,24(11):2721−2733. http://www.jos.org.cn/1000-9825/4478.htm [doi: 10.3724/SP.J.1001.2013.04478]

印鉴(1968－),男,湖北仙桃人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据,数据挖掘.
E-mail: issjyin@mail.sysu.edu.cn

李琪(1991－),女,硕士生,主要研究领域为推荐系统,社会网络.
E-mail: qiqiaisheng@gmail.com

王智圣(1986－),男,博士生,主要研究领域为大数据,推荐系统.
E-mail: zhishwang@gmail.com

苏伟杰(1986－),男,硕士生,主要研究领域为数据挖掘,推荐系统.
E-mail: swj222225@126.com