大数据环境下基于概率矩阵分解的个性化推荐

田贤忠 沈 杰
(浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 概率矩阵分解是近几年广泛应用的协同过滤推荐方法。针对如何利用矩阵分解技术提高推荐质量以及在大数据环境下如何突破计算时间、计算资源瓶颈等问题进行研究,提出了 Improved Probabilistic Matrix Factorization (IPMF)融入邻居信息的概率矩阵分解算法,并且提出了 parallel-IPMF (p-IPMF)算法来解决融入邻居信息后计算复杂度高和难以并行化等问题。在 MapReduce 并行计算框架下将 p-IPMF 算法加以实现,并在真实数据集上进行验证。实验结果表明,所提算法能有效提高推荐质量并缩短计算时间。

关键词 推荐算法,概率矩阵分解,大数据,MapReduce
中图法分类号 TP393 **文献标识码** A

Personalized Recommendation Based on Probabilistic Matrix Factorization in Big Data Environment

TIAN Xian-zhong SHEN Jie
(School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract Probabilistic matrix factorization is a type of collaborative filtering algorithm which is widely used in recent years. Based on the problem of how to use matrix factorization technology to improve the recommendation quality and how to breakthrough the limitation of calculation time and resource in big data environment, we introduced an improved probabilistic matrix factorization algorithm which integrates neighbor information and introduced parallel-IPMF, overcoming the problem of high calculation complex and the problem of parallelization. We used the real dataset to implement our algorithm on the MapReduce parallel computation framework. The experiment results show that our algorithm can improve the recommendation quality and reduce the computation time.

Keywords Recommendation algorithm, Probabilistic matrix factorization, Big data, MapReduce

1 引言

近年来,信息的爆炸式增长和用户个性化需求的提高不断推动着推荐系统的研究与发展。个性化推荐算法通常分为基于内容的推荐算法、基于关联规则的推荐算法、基于协同过滤的推荐算法以及混合推荐算法[1]。其中,基于协同过滤的推荐算法受到业界广泛好评并被广泛应用。

基于潜在因子模型的推荐技术是一类更加精确的协同过滤推荐技术,概率矩阵分解[2]是其常用的一种方法。该模型把用户、产品特征映射到低维空间,通过拟合已知的评分信息计算潜在的特征向量,从而构造出用户-项目评分矩阵。矩阵分解模型具有较高的推荐精度和扩展性。因此,Salakhutdnov 等人[2]提出的概率矩阵分解模型(Probabilistic Matrix Factorization, PMF)得到了广泛的应用与研究。张志军等人[3]把社交网络中的信任关系与项目消费时序因素分别融入到 PMF 模型的用户矩阵与项目矩阵,以扩展原有模型;刘强等人[4]提出了融入物品情景信息的矩阵分解模型。这些工作都对提高推荐精度有一定帮助,但是将额外的信息融入模型也大大增加了算法的复杂度和计算时间。随着大数据时代的到来,推荐系统需要处理海量的数据,计算时间、计算资源成为了瓶颈,很多基于并行化矩阵分解的研究工作应运而生。Imen Chakroun 等人[5]研究基于 GPU 并行化计算代价高昂的贝叶斯概率矩阵分解模型。Gemulla 等人[6-7]致力于研究并行使用随机梯度下降法,以求解矩阵分解最终的优化问题,对矩阵采取分块以避免并行重写冲突;印鉴等人[8]通过 MapReduce[9]计算框架并行化随机梯度下降法,并利用隐式反馈数据获得推荐模型;王全民等人[10]致力于研究通过并行化迭代交替最小二乘法求解矩阵分解优化问题。上述工作都对矩阵分解模型的并行计算做出了贡献,但是所做工作都没有涉及到用户矩阵或者项目矩阵在融入其他信息后该如何并行化求解优化问题。

本文针对矩阵分解模型在揭示用户之间的互相作用关系时效果不理想、并行化计算融入额外信息的矩阵分解模型困难等问题,给出了一种解决方案。通过聚类技术获得用户近邻信息,在矩阵中融入近邻信息,根据聚类结果对用户、项目进行分组,调整更新序列,在 MapReduce 计算框架下实现并行计算。主要贡献如下:1)定义用户间对项目偏好的相似度由两部分组成:用户对项目的评分相似度以及用户对不同项

本文受国家自然科学基金项目(61672465),浙江省自然科学基金项目(LY15F020027)资助。
田贤忠(1968-),男,博士,教授,主要研究方向为无线网络、网络编码和大数据,E-mail:txz@zjut.edu.cn;沈 杰(1991-),男,硕士,主要研究方向为大数据、推荐系统。

目类别的关注度；2）根据相似度对用户进行聚类，并把用户近邻信息融入到概率矩阵分解模型；3）根据聚类结果对用户、项目进行分组，调整更新序列，在 MapReduce 计算框架下实现并行计算。

本文第 2 节介绍本文融入用户近邻信息的矩阵分解模型；第 3 节介绍算法基于 Mapreduce 模型的并行化实现；第 4 节给出实验并对结果进行分析；最后总结全文并指出未来的工作。

2　融入用户近邻信息的改进矩阵分解模型 IPMF

2.1　聚类

传统协同过滤技术中常用 KNN(K-Nearest Neighbor)获得用户近邻信息。本文为解决 KNN 方法复杂度过高的问题，同时为并行化融入用户的近邻信息矩阵分解模型做准备，采用聚类技术聚类用户，进而获得用户的近邻信息。

首先定义用户之间对项目偏好的相似度，该相似度由两部分组成：用户对项目的评分相似度；用户对项目不同类别的关注相似度。项目评分相似度能够表现用户对单个项目的喜好，而项目类别关注相似度能够体现用户对这个项目类别的喜好。存在用户对某项目评价很低但是却对这个项目类目很感兴趣的情况，这时向其推荐该类目下的其他项目也是合理的；反之亦然。因此，这两种相似度是应该综合考虑的。

设用户集 $U=\{u_1,u_2,\cdots,u_n\}$，项目集 $I=\{i_1,i_2,\cdots,i_m\}$，I_c 表示用户 u 和用户 v 共同评过分的项目集合，r_{ui} 表示用户 u 对项目 i 的评分，$\bar{r}_u$ 表示用户 u 在项目集合 I_c 下的平均评分。本文采用皮尔逊相关相似度计算来用户对项目的评分相似度，记为 $sim(u,v)_a$：

$$sim(u,v)_a=\frac{\sum_{i\in I_c}(r_{ui}-\bar{r}_u)(r_{vi}-\bar{r}_v)}{\sqrt{\sum_{i\in I_c}(r_{ui}-\bar{r}_u)^2\sum_{i\in I_c}(r_{vi}-\bar{r}_v)^2}} \tag{1}$$

用余弦相似度来计算用户对项目不同类别的关注相似度，记为 $sim(u,v)_b$：

$$sim(u,v)_b=\frac{\sum_{c\in C}(n_{uc}*n_{vc})}{\sqrt{\sum_{c\in C}n_{uc}^2}*\sqrt{\sum_{c\in C}n_{vc}^2}} \tag{2}$$

其中，$C=\{c_1,c_2,\cdots,c_l\}$ 表示项目类别，n_{uc} 表示用户 u 在项目类别 c 下的评分个数。根据式(1)、式(2)，可得用户对项目的偏好相似度 $sim(u,v)$ 为：

$$sim(u,v)=\alpha sim(u,v)_a+(1-\alpha)sim(u,v)_b \tag{3}$$

其中，α 为平衡因子。

本文采用 K-means 方法对用户聚类，步骤如下：

1. 随机选取 k 个用户作为聚类中心 C=[c_1,…,c_k]。
2. For each u∈U do
   2.1 根据式(3)计算用户 u 和所有聚类中心的相似度，选择相似度最大的类 c。
   2.2 将用户 u 加入类 c，更新类中心(求出类中所有用户对项目评分的平均值和所有用户在各项目类别下的平均评分数目)。
3. 若中心点改变，则返回步骤 2；否则聚类完成，返回 k 个聚类。

在聚类完成后，仅在用户 u 所属聚类中寻找 u 的项目偏好相似近邻，根据式(3)找出 N 个最相似邻居，N 为自定义的近邻数。

2.2　概率矩阵分解模型

本文的推荐算法主要是基于用户-项目评分矩阵进行相应的计算，推荐系统的目的是预测用户对未评分项目的评分。具体而言，假设在推荐系统中存在 N 个用户和 M 个项目，其构成的集合为用户集 $U=\{u_1,u_2,\cdots,u_n\}$，项目集 $I=\{i_1,i_2,\cdots,i_m\}$，用户-产品评分矩阵 $R=[r_{ui}]_{N\times M}$，在这个评分矩阵中 r_{ui} 表示用户 u 对产品 i 的评分(比如 1～5 分)。概率矩阵分解模型学习用户(项目)的潜在特征向量，然后基于此特征向量预测未知的评分。假设 $U\in R^{K\times M}$ 和 $V\in R^{K\times N}$ 代表用户和产品的特征矩阵，其中 U_u 和 V_i 分别表示某个特定用户 u 和项目 i 的 K 维特征向量。一般来说，维度越高越能刻画特征，但是维度越高复杂度也越高，并且当维数达到一定值后，再增加维度也不能明显提高推荐精度。

概率矩阵分解模型赋予用户和项目的隐式特征向量以均值为 0 的高斯先验，σ_U^2 和 σ_V^2 分别为用户和项目特征的向量方差，I 为单位矩阵。

$$p(U|\sigma_U^2)=\prod_{u=1}^{N}N(U_u|0,\sigma_U^2I) \tag{4}$$

$$p(V|\sigma_V^2)=\prod_{i=1}^{M}N(V_i|0,\sigma_V^2I) \tag{5}$$

假设已观测到的评分数据的条件概率如下：

$$p(R|U,V,\sigma_R^2)=\prod_{u=1}^{N}\prod_{i=1}^{M}[N(R_{ui}|U_u^{\mathrm{T}}V_i,\sigma_R^2)]^{I_{ui}^R} \tag{6}$$

其中，I_{ui}^R 在用户 u 对项目 i 有评分时取 1，否则取 0。通过贝叶斯推导，可知用户和项目的隐式特征的后验概率的取值：

$$p(U,V|R,\sigma_U^2,\sigma_V^2,\sigma_R^2)=\propto p(R|U,V,\sigma_R^2)\times p(U|\sigma_U^2)\times p(V|\sigma_V^2) \tag{7}$$

其目的是使隐式特征后验概率最大。为方便求导，对其取自然对数：

$$\begin{aligned}\ln\ & p(U,V|R,\sigma_U^2,\sigma_V^2,\sigma_R^2)\\ =&-\frac{1}{2\sigma_R^2}\sum_{u=1}^{N}\sum_{i=1}^{M}I_{ui}^R(R_{ui}-U_u^{\mathrm{T}}V_i)^2-\frac{1}{2\sigma_U^2}\sum_{u=1}^{N}U_u^{\mathrm{T}}U_u-\\ &\frac{1}{2\sigma_V^2}\sum_{i=1}^{M}V_i^{\mathrm{T}}V_i-\frac{1}{2}[(\sum_{u=1}^{N}\sum_{i=1}^{M}I_{ui}^R)+N\times K\ln\sigma_U^2+M\times K\ln\sigma_V^2]+C\end{aligned} \tag{8}$$

上式取最大，即最小化如下目标函数：

$$F=\frac{1}{2}\sum_{u=1}^{N}\sum_{i=1}^{M}I_{ui}^R(R_{ui}-U_u^{\mathrm{T}}V_i)^2+\frac{\sigma_R^2}{2\sigma_U^2}\sum_{u=1}^{N}\|U_u\|^2+\frac{\sigma_R^2}{2\sigma_V^2}\sum_{i=1}^{M}\|V_i\|^2 \tag{9}$$

在融入用户近邻信息的概率矩阵分解模型中，用户 u 的特征向量可以表示为其邻居向量的加权和，N_u 表示用户 u 的邻居用户，S_{ui} 表示用户 u 与用户 i 的项目偏好相似度，该相似度由式(3)计算可得。对 S_{ui} 进行归一化处理：

$$\sum_{i\in N_u}S_{ui}=1 \tag{10}$$

$$U_u=\sum_{i\in N_u}U_iS_{ui} \tag{11}$$

与基本概率矩阵分解模型不同，本文引入了一个新的条件概率：

$$p(U|\sigma_U^2,\sigma_S^2)=p(U|\sigma_U^2)\times p(U|S,\sigma_S^2) \tag{12}$$

其中，$p(U|S,\sigma_S^2)$ 表示用户在已知近邻关系的条件下其特征向量的概率，且该概率服从如下高斯分布：

$$p(U|S,\sigma_S^2)=\prod_{u=1}^{N}N(U_u|\sum_{i\in N_u}U_iS_{ui},\sigma_S^2I) \tag{13}$$

在此改进基础上，本文得到用户、项目新的特征向量后验概率：

$$p(U,V|R,\sigma_U^2,\sigma_V^2,\sigma_R^2)=\propto p(R|U,V,\sigma_R^2)\times p(U|\sigma_U^2)\times p(U|S,\sigma_S^2)\times p(V|\sigma_V^2) \tag{14}$$

获得改进后的目标函数：

$$F=\frac{1}{2}\sum_{u=1}^{N}\sum_{i=1}^{M}I_{ui}^{R}(R_{ui}-U_u^{\mathrm{T}}V_i)^2+\frac{\sigma_R^2}{2\sigma_U^2}\sum_{u=1}^{N}\|U_u\|^2+\frac{\sigma_R^2}{2\sigma_V^2}\sum_{i=1}^{M}\|V_i\|^2+\frac{\sigma_R^2}{2\sigma_S^2}\sum_{u=1}^{N}[(U_u-\sum_{p\in N_u}U_pS_{up})^{\mathrm{T}}(U_u-\sum_{p\in N_u}U_pS_{up})] \tag{15}$$

本文采用随机梯度下降法求取以上目标函数的最小解，分别获得如下用户、项目特征偏导公式：

$$\frac{\partial F}{\partial U_u}=\sum_{i=1}^{M}I_{ui}^{R}V_i(U_u^{\mathrm{T}}V_i-R_{ui})+\lambda_U U_u+\lambda_S(U_u-\sum_{p\in N_u}U_pS_{up})-\lambda_S\sum_{p|u\in N_p}(U_p-\sum_{x\in N_p}U_pS_{px}) \tag{16}$$

$$\frac{\partial F}{\partial V_i}=\sum_{u=1}^{N}I_{ui}^{R}U_u(U_u^{\mathrm{T}}V_i-R_{ui})+\lambda_V V_i \tag{17}$$

其中，$\lambda_U=\frac{\sigma_R^2}{\sigma_U^2}$，$\lambda_S=\frac{\sigma_R^2}{\sigma_S^2}$，$\lambda_V=\frac{\sigma_R^2}{\sigma_V^2}$为惩罚系数，防止用户和项目矩阵过拟合。至此，个性化推荐问题已被转化为最优化问题。通过计算得到用户、项目特征向量预测用户对项目的评分，进而可以根据评分由高到低对项目进行推荐。

算法 1 IPMF 算法

输入：用户-项目评分矩阵 R，用户相似矩阵 S

输出：用户特征矩阵，项目特征矩阵

```
1. 初始化用户特征矩阵，项目特征矩阵
2. for each u∈U do//对每个用户批量处理其评分数据，更新特征向量
3.   for k=1,2,…,K do //在每个特征维度上获得邻居用户对特征向量的影响
4.     for each p∈N_u do
5.       s_1k+=U_pk S_up
6.       for each x∈N_p do
7.         s_2k+=U_xk S_px
8.       end for
9.       y_k=U_pk-s_2k
10.    end for
11.    x_k=U_uk-s_1k
12.  end for
13.  for each r_ui∈R//遍历评分信息
14.    e=r_ui-U_u V_i
15.    for k=1,2,…,K do//在各维度更新特征向量值
16.      U_uk-=α*(e*V_ik+λ_U*U_uk+λ_S*x_k-λ_S*y_k)
17.      V_ik-=α*(e*U_uk+λ_V*V_ik)
18.    end for
19.  end for
20. end for
```

在算法中为避免每次处理用户 u 的不同评分记录都需要获取其邻居信息，在处理用户 u 时批量处理其所有的评分记录。这样虽然不是严格意义上的随机梯度下降法，但是可以避免重复计算用户的邻居带来的特征向量的影响，该调整对模型最终的推荐效果的影响并不大。

2.3 算法复杂度分析

在采用随机梯度下降法求解式(9)时，只需要根据评分信息更新用户、项目特征向量。每一次迭代更新的复杂度为 $O(K)$，K 为特征向量的维度。但是，当为了揭示用户之间相互作用关系，提高推荐质量，进一步融入用户的近邻信息时，由于需要考虑该用户的邻居和该邻居的邻居信息，求解式(15)时算法的复杂度剧增为 $O(N^2K)$，N 为用户近邻个数。当面临海量数据时，训练模型的时间将非常长，计算能力成为瓶颈，并行化计算成为解决此类问题的方案。

3 并行化推荐模型 p-IPMF 及其 MapReduce 实现

3.1 并行化推荐模型

并行化处理随机梯度下降法(Stochastic Gradient Descent，SGD)具有以下难点：1)SGD 每一步更新用户(项目)特征向量须依赖上一步的更新值，即并行的每个处理器必须知道最新的用户(项目)特征向量；2)SGD 需要避免重写问题，即并行的处理器避免同时更新相同的特征向量；在融入用户的近邻信息后，处理器在更新当前用户的特征向量时不仅需要上一步中该用户的特征向量，还需要其邻居的上一步特征向量。

由此给出并行解决方案：前期对用户进行聚类，使得任何用户与邻居用户属于同一类，这一做法可以保证用户获得其邻居用户的最新特征向量；进而根据聚类结果对用户分组，并对项目进行随机分组。组合用户、项目分组，对不同的组合在不同的处理节点并行计算，这一做法可以保证在一次更新作业中一个用户或项目的特征向量只会在一个计算更新。

下面结合图 1 和图 2 进一步阐述执行过程。设用户通过聚类分成 6 类，再根据聚类结果随机分成 3 组，项目随机分成 3 组，如图 1 所示。

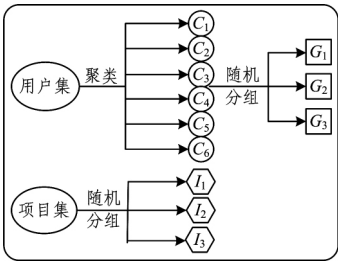

图 1 用户、项目分组示例

设有 3 个处理节点，在将用户、项目集合分组后，再把分组两两组合，使得组合之间没有交集，把组合划分到各计算节点，分别在每个节点上处理组合数据集。如图 2 所示，节点 A，B，C 并行处理分配给它的组合数据集，通过调整更新序列可以达到各节点独立而不冲突地更新用户、项目特征向量的目的。

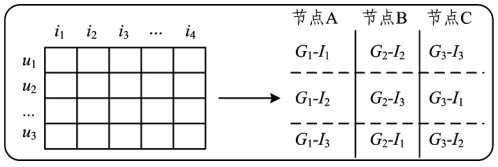

图 2 并行处理示例

3.2 MapReduce 实现

在 MapReduce 模型中，计算过程主要有 Map 和 Reduce 两阶段。两个阶段的输入、输出是〈key，value〉的键值对，定

义好键值对和利用阶段之间的 Shuffle 过程是该计算模型的关键。

文中 Map 阶段负责读入数据，获取用户分组和项目分组，并且确定用户、项目组合分组 id，并将其作为键值对中的 key。value 为用户 id、项目 id 和用户对项目的评分。Shuffle 过程把分散在各 Map 节点的键值对数据按照组合分组 id 划分到相应的 Reduce 节点。Reduce 阶段负责求解优化问题，输出键值对项目 id 或用户 id 以及特征向量。下面进一步给出具体的算法。

(1)Map 阶段

输入：用户-项目评分信息，包含用户 id、项目 id、评分(1－5)

输出：〈用户-项目组合 id，〈用户 id，项目 id，评分〉〉

1. 获取用户分组、项目分组、用户-项目组合信息
2. 处理每行输入数据，获得用户 id、项目 id、评分
3. 把该评分记录划分到相应的用户-项目分组组合中，输出〈用户-项目组合 id，〈用户 id，项目 id，评分〉〉

(2)Reduce 阶段

输入：〈用户-项目组合 id，列表 L〈用户 id，项目 id，评分〉〉

输出：〈用户或项目 id，特征向量〉

1. 获取最新用户、项目特征向量
2. for each 〈用户 id，项目 id，评分〉∈L
3. 按照单机处理算法更新用户、项目特征向量
4. end for
5. 输出〈用户或项目 id，特征向量〉

Map 端处理比较简单，主要为数据的映射划分、过滤不完整数据等。Reduce 端负责实现并行更新用户、项目特征向量。可以观察到，Reduce 需要获取最新用户、项目特征向量，此类操作可以在 Reduce 端执行 reduce 函数前完成装载，最终输出的特征向量作为下一迭代的输入，当满足收敛条件停止迭代，并保存最终结果。

4　实验结果及分析

本文实验部署在 Cloudera[11] 发行的 Hadoop(Hadoop 2.6.0-cdh5.6.0)集群。该集群由 4 台 16G 内存、AMD Opteron 8 核处理器主机组成，其中 1 台做主节点协调，3 台计算节点。采用来自 Minnesota 大学 GroupLens 项目组的 MovieLens 实验数据集。MovieLens 数据集包含 6040 个用户、3900 部电影、1000209 次评分(1～5)，其稀疏度为6.31%，随机抽取其中 80% 的数据作为训练集，20% 作为测试集，重复实验 5 次。

衡量推荐系统推荐精度的标准有多种，本文采用被广泛使用的均方根误差 RMSE 进行度量。RMSE 值越小，推荐精度越高。

$$RMSE=\sqrt{\frac{1}{|E^p|}\sum_{(u,i)\in E^p}(r_{ui}-\hat{r}_{ui})^2} \tag{18}$$

其中，E_p 表示测试集，$|E_p|$ 表示测试集中用户对项目评分的个数；r_{ui} 表示用户 u 对项目 i 的真实评分；$\hat{r}_{ui}$ 表示用户 u 对项目 i 的预测评分。

(1)潜在特征向量维度对推荐精度的影响

增加潜在特征向量的维度会提高推荐精度，而过高的维度会导致计算耗时太长。为找到一个平衡点，设计一组自变量为特征向量维度、因变量为推荐精度的实验结果，并且实验引入模型训练的迭代次数。在不同的迭代次数下观察实验结果。

如图 3 所示，5 条曲线为本文算法在潜在特征向量维度为 5～25 上的实验结果，横坐标为迭代次数，纵坐标为误差值。

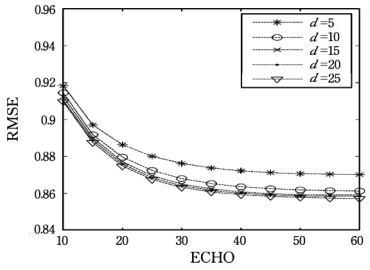

图 3　维度对推荐精度的影响

可以观察到：维度由 5 增加到 10 时，RMSE 下降明显，推荐精度提高较多；维度由 10 增加到 15 时，精度提高较少；而维度 15，20，25 的实验曲线已经接近重合，表明即使再增加维度，对提高推荐精度也已经没有明显作用。而维度高影响计算时间，因此本文以下实验选取维度为 20。

(2)推荐精度对比

为验证本文算法确实能够提高推荐精度，引入传统的概率矩阵分解模型(PMF)和基于用户近邻的协同过滤推荐技术(CF)来与本文算法做推荐精度上的对比实验。

图 4 所示为本文算法 IPMF 与传统的 PMF 和 CF 算法在不同迭代次数上的 RMSE 值比较。在不同迭代次数上，RMSE 值一直满足 IPMF＜PMF＜CF，当迭代次数大于 50 时，RMSE 都没有明显变化。此实验中，本文算法平衡因子 α 取值为 0.5；参数 $\lambda_U=\lambda_V=0.12$，$\lambda_S=0.1$；学习步伐初始值为 0.04，步伐值随着迭代次数的增加而逐渐减小。

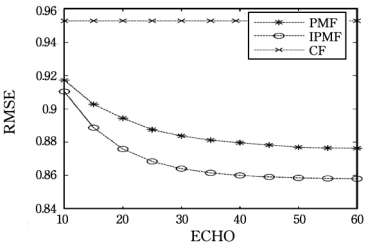

图 4　推荐精度对比

(3)在 MR 上实现 p-IPMF，对比计算时间

本文算法在 Hadoop 平台实现并行计算。为验证并行计算在增加计算节点后也能有效减少计算时间，设计一组自变量为计算节点数、因变量为计算时间的对比实验。

图 5 所示为本文算法在 Hadoop 平台的 MapReduce 实现。分布式系统由于有额外的协调开销，计算时间虽然不是呈比例减少，但也随着计算节点数的增加而明显减少。

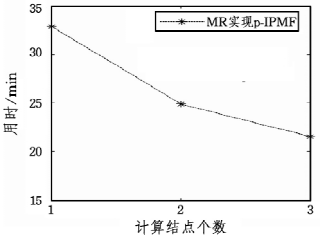

图 5　执行时间的对比

(下转第 469 页)

结束语　随着存储系统中数据量的急剧增加和冗余编码在存储系统中越来越广泛地被应用，存储失效导致的数据重建问题越来越引起冗余编码领域的重视。本文对近年提出的减少重建数据量的冗余编码进行了系统分析，从编码的实现原理、基于冗余编码系统的特性与开销和不同冗余编码的编码性能方面进行了讨论，为不同减少重建数据量的冗余编码在实际系统中的应用提供了参考。

参 考 文 献

[1] BORTHAKUR D, SCHMIDT R, VADALI R, et al. Hdfs raid [C]// Hadoop User Group Meeting. 2010.

[2] WEIL S A, BRANDT S A, MILLER E L, et al. Ceph: A scalable, high-performance distributed file system[C]// Proceedings of the 7th Symposium on Operating Systems Design and Implementation. USENIX Association, 2006: 307-320.

[3] PLANK J S, DING Y. Note: Correction to the 1997 tutorial on Reed-Solomon coding[J]. Software: Practice and Experience, 2005, 35(2): 189-194.

[4] KHAN O, BURNS R C, PLANK J S, et al. Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads[C]// FAST. 2012: 20.

[5] DIMAKIS A G, GODFREY P, WU Y, et al. Network coding for distributed storage systems[J]. IEEE Transactions on Information Theory, 2010, 56(9): 4539-4551.

[6] SATHIAMOORTHY M, ASTERIS M, PAPAILOPOULOS D, et al. Xoring elephants: Novel erasure codes for big data[J]. Proceedings of the VLDB Endowment, VLDB Endowment, 2013, 6(5): 325-336.

[7] RASHMI K V, SHAH N B, GU D, et al. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster[C]// Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems. 2013.

[8] SHAH N B, RASHMI K V, KUMAR P V, et al. Interference alignment in regenerating codes for distributed storage: Necessity and code constructions[J]. IEEE Transactions on Information Theory, 2012, 58(4): 2134-2158.

[9] RASHMI K V, NAKKIRAN P, WANG J, et al. Having your cake and eating it too: jointly optimal erasure codes for I/O, storage, and network-bandwidth[C]// 13th USENIX Conference on File and Storage Technologies (FAST 15). 2015: 81-94.

[10] HU Y, CHEN H C H, LEE P P C, et al. NCCloud: applying network coding for the storage repair in a cloud-of-clouds[C]// FAST. 2012: 21.

[11] HUANG C, SIMITCI H, XU Y, et al. Erasure coding in windows azure storage[C]// Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12). 2012: 15-26.

[12] PAPAILIOPOULOS D S, LUO J, DIMAKIS A G, et al. Simple regenerating codes: Network coding for cloud storage[C]// 2012 Proceedings IEEE INFOCOM. IEEE, 2012: 2801-2805.

[13] LIU Q, FENG D, JIANGY H, et al. Z Codes: General Systematic Erasure Codes with Optimal Repair Bandwidth and Storage for Distributed Storage Systems[C]// 2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS). IEEE, 2015: 212-217.

[14] PLANK J S, XU L. Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications[C]// Fifth IEEE International Symposium on Network Computing and Applications, 2006(NCA 2006). IEEE, 2006: 173-180.

[15] PLANK J S, GREENAN K M, MILLER E L. Screaming fast Galois field arithmetic using intel SIMD instructions[C]// FAST. 2013: 299-306.

[16] HUANG C, XU L. STAR: An efficient coding scheme for correcting triple storage node failures[J]. IEEE Transactions on Computers, 2008, 57(7): 889-901.

（上接第 441 页）

结束语　本文针对提高矩阵分解技术的推荐质量和在大数据环境下突破计算时间、计算资源瓶颈等问题进行研究，提出了融入邻居信息的矩阵分解算法，并且克服了融入邻居信息后计算复杂度高和难以并行化等问题。最后，在 MapReduce 并行计算框架下实现了所提并行计算。

参 考 文 献

[1] ADOMAVICIUS G, TUZHILIN A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6): 734-749.

[2] SALAKHUTDINOV R, MNIH A. Probabilistic Matrix Factorization[M]// Advances in neural information processing systems, NIPS'08. Cambridge, Massachusetts, USA, MIT Press, 2008: 1257-1264.

[3] ZHANG Z J, LIU H. Social Recommendation Model Combining Trust Propagation and Sequential Behaviors[C]// Applied Intelligence. 2015.

[4] LIU Q, WANG C W, XU C F. A modified PMF modelincorporating implicit item associations[C]// Proceedings of 24th International Conference on Tools with Artificial Intelligence. Athens, Greece, 2012.

[5] CHAKROUN I, Haber T, AA T V. Exploring Parallel Implementations of the Bayesian Probabilistic Matrix Factorization [C]// Parallel, Distributed, and Network-Based Processing (PDP). 2016.

[6] GEMULLA R, HAAS P J, NIJKAMP E, et al. Large-Scale matrix factorization with distributed stochastic gradient descent [C]// Proc. of the 17th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2011: 69-77.

[7] RECHT B, RÉ C. Parallel stochastic gradient algorithms for large-scale matrix completion[J]. Mathematical Programming Computation, 2013, 5(2): 201-226.

[8] 印鉴，王智圣，李琪，等. 基于大规模隐式反馈的个性化推荐[J]. 软件学报，2014，25(9)：1953-1966.

[9] LÄmmel R. Google's MapReduce programming model-revisited [J]. Science of Computer Programming, 2007, 68(3): 208-237.

[10] 王全民，苗雨，何明，等. 基于矩阵分解的协同过滤算法的并行化研究[J]. 计算机技术与发展，2015，25(2)：55-59.

[11] Cloudera. Cloudera[EB/OL]. http://www.cloudera.com.