计算机科学
COMPUTER SCIENCE

基于点割集图分割的矩阵变换与分解的推荐算法

何亦琛，毛宜军，谢贤芬，古万荣

引用本文

何亦琛，毛宜军，谢贤芬，古万荣. 基于点割集图分割的矩阵变换与分解的推荐算法[J]. 计算机科学, 2022, 49(6A): 272-279.

HE Yi-chen, MAO Yi-jun, XIE Xian-fen, GU Wan-rong. Matrix Transformation and Factorization Based on Graph Partitioning by Vertex Separator for Recommendation[J]. Computer Science, 2022, 49(6A): 272-279.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210600159

基于点割集图分割的矩阵变换与分解的推荐算法

何亦琛[1]　毛宜军[1]　谢贤芬[2]　古万荣[1]
1 华南农业大学数学与信息学院　广州 510642
2 暨南大学经济学院　广州 510632
(heyichen666@hotmail.com)

摘　要　基于模型的协同过滤算法通过矩阵分解来将用户偏好以及物品属性用隐变量来表示,但现有的矩阵分解算法很难应对个性化推荐系统中严重的数据稀疏性以及数据变化性所带来的问题。针对上述问题,提出了基于双边块对角矩阵的矩阵分解算法。首先通过基于社区发现的点割集图分割算法将原始的稀疏矩阵转换成双边块对角矩阵,将具有相同偏好的用户以及相似特征的物品归并到同一个对角块中,然后将结构中的对角块和双边拼接成数个密度较高的子矩阵。实验结果表明,对这几个密度有提高的子矩阵进行并行分解,基于其分解结果进行原始矩阵的预测,能够有效缓解矩阵分解中数据稀疏性所带来的问题,既能提升预测的精度,又能提高推荐结果的可解释性。同时,每个子对角块都能并行独立分解,能达到提高算法效率的目的。

关键词:社区发现;矩阵分解;协同过滤;推荐系统
中图法分类号　TP391.3

Matrix Transformation and Factorization Based on Graph Partitioning by Vertex Separator for Recommendation

HE Yi-chen[1],MAO Yi-jun[1],XIE Xian-fen[2] and GU Wan-rong[1]
1 School of Mathematics and Information,South China Agricultural University,Guangzhou 510642,China
2 School of Economy,Jinan University,Guangzhou 510632,China

Abstract　Model-based collaborative filtering algorithms usually express user's preferences and item's attributes by latent factors through matrix factorization,but the traditional matrix factorization algorithm is difficult to deal with the serious data sparsity and data variability problems in the recommendation system. To solve the above problem,a matrix factorization algorithm based on bordered block diagonal matrix is proposed. Firstly,the original sparse matrix is transformed into bordered block diagonal matrix by a graph partitioning algorithm based on community discovery,which merges users with the same preference and items with similar characteristics into the same diagonal block,and then splices the diagonal blocks and the bordered into several sub-diagonal matrices which have higher densities. The experimental results show that,by decomposing the sub-diagonal matrices in parallel can not only improve the precision of prediction,but also improve the interpretability of the recommendation results. At the same time,each sub-diagonal matrix can be decomposed independently and in parallel,which can improve the efficiency of the algorithm.

Keywords　Community discovery,Matrix factorization,Collaborative filtering,Recommendation system

1　引言

随着互联网的高速发展,人们在线上的活动也变得越来越频繁,因此产生了庞大且复杂的数据集。如何利用这些庞大的数据集将个性化推荐应用到不同的场景是当下需要解决的热点问题。

协同过滤算法充分利用集体的智慧,通过分析大量用户的行为来挖掘某些隐含的模式,基于协同过滤的推荐算法得到了最广泛以及最深入的研究[1-2]。基于协同过滤的推荐系统又可以细分为基于用户的协同过滤系统、基于物品的协同过滤系统、基于模型的协同过滤系统[3-5]。其中,基于隐变量模型的协同过滤是最为高效的协同过滤算法之一[6],基于矩阵分解的协同过滤算法将用户偏好用少量的隐变量来表示,矩阵分解算法的思想都是将原始矩阵分解为子矩阵,利用子矩阵的乘积对原始矩阵进行还原及预测。常见的矩阵分解算法包括 SVD(奇异值分解)[7]、NMF(非负矩阵分解)[8]、

基金项目:全国统计科学研究项目(2020LY018);全国统计科学研究重点项目(2019LZ37);广东省社会科学项目(GD19CGL34);国家重点研发计划(2017YFC1601701);广东省农业厅创新团队项目(2019KJ130)
This work was supported by the National Statistical Science Research Project(2020LY018),National Statistical Science Research Key Project(2019LZ37),Social Science Project of Guangdong Province(GD19CGL34),National Key Research and Development Project(2017YFC1601701) and Department of Agriculture of Guangdong Provincial Innovation Research Team Project(2019KJ130).
通信作者:古万荣(guwanrong@scau.edu.cn)

PMF(概率矩阵分解)[9]等等。

目前的矩阵分解算法虽然在推荐系统应用的预测精度方面有着较好的表现，但是依旧面临着以下问题：

(1)数据的稀疏性，在实际的系统中，通常用户只对系统中极个别的物品进行过评分，若直接对整个原始矩阵进行矩阵分解，则必然会带来计算量庞大的问题。

(2)数据的变化性，在实际的系统中，用户的行为数据会不断变化，模型的局部变化必然会对整个模型带来改变，如何对模型进行重复训练是需要解决的问题。

(3)推荐结果的可解释性，矩阵分解的分解结果是基于隐变量的，但并不能很明确地确定这些隐变量的具体含义，很难给予用户一个可接受且信服的推荐结果。

为解决上述问题，本文提出了基于双边块对角矩阵的矩阵分解算法，主要贡献可以总结如下：

(1)受社区发现算法启发，本文利用基于社区发现的点割集图分割算法，对原始稀疏矩阵进行划分，得到双边块对角矩阵结构，该结构将具有相似偏好的用户和具有相似属性的物品聚集在一起，提升原始矩阵的密度，缓解数据稀疏性带来的问题，基于该结构进行矩阵分解并进行协同过滤，也可以对推荐结果进行更好的解释。

(2)本文提出了基于上述结构的近似矩阵分解算法，实验结果表明，相比 4 种基线算法，本文提出的基于双边块对角矩阵的矩阵分解算法的预测精度以及速度都有较大幅度的提高。同时，当原始矩阵的某处评分发生变化后，不再需要对整个模型进行重新分解，只需要在发生变换的子矩阵上重新分解即可，从而大大提升了在实际推荐系统的分解效率以及可扩展性。

2 相关工作

为了解决数据稀疏性的问题，目前较为流行的方法为矩阵聚类算法[10-12]，该算法预先将用户的行向量以及物品的列向量进行聚类，然后根据聚类的结果，将预测限定在特定的类别之中，以达到降低预测计算量的目的。但是聚类算法也有以下不足之处，通常来说，聚类的结果难以给出令用户信服的解释，并且聚类算法将系统中的一个用户或一个物品强制归属于一个类中，这与实际的系统也是相违背的。另一种数据降维方法 PCA 方法(主成分分析)是以 SVD 为基础发展起来的方法，PCA 方法的思想是把用户和物品映射到另外一个潜在特征空间中，在这个空间中，用户和物品的最主要特征可以很轻易地表达出来。已有学者，如 Aaldering[13]直接将 PCA 应用到协同过滤之中。但是，PCA 方法也有如下的缺点：PCA 方法无法通过参数化等方法对处理过程进行干预，假设系统开发人员已对需要进行预测的内容有一定的先验知识，但是无法调节其方法，这样可能导致预测效果不理想，同时算法效率也会受到影响。

为了解决模型需要重复训练的问题，学者们对流行的矩阵分解算法进行了拓展研究，如 Nguyen[14]提出了一种分布式机器学习方法(DE-SVD)，该算法使用边缘服务器进行数据初步处理，数据的最终处理则由集中式服务器完成。还有 Yu 等[15]提出使用坐标下降法代替交替二乘法(ALS)和随机梯度下降法(SGD)去计算矩阵分解问题的最优解。Liu 等[16]提出了基于 SVD 填充的混合推荐算法，也通过随机梯度法去填充稀疏矩阵，并且将时间权重融入矩阵中以优化用户相似度。他们的目的都是为了解决在频繁交互下模型可扩展性不足的问题。但是这些算法在矩阵的局部发生变化时，仍需要对整个模型进行分解，在数据量庞大的情况下，其计算量依旧非常大。

为了更好地对用户和物品进行分类，并给出令用户信服的推荐结果，社区发现算法开始得到了学者们的研究[17-20]。聚类算法和社区发现算法具有相似之处，但是两种算法处理的对象不同，聚类算法所处理的是能表示为高维向量的属性数据，而社区发现算法所处理的是能表示为网络的关系数据。近来，也有不少学者将社区发现算法与矩阵分解算法结合起来进行研究，如 Gligorijević 等[21]将 NMF 算法和社区发现结合起来进行研究，提高了矩阵分解结果的可解释性。

3 基于社区发现的点割集图分割算法

对于无向图 $G(V,E)$，有顶点集 $V'\subset V$，$V''\subset V'$，如果删除 V'，则 G 不再连通，但如果删除 V' 的子集 V'' 后图依旧是连通图，则称 V' 是无向图 G 的一个点割集。

每一个稀疏的矩阵都可以用一个二部图来表示(见图 1)，假设对原始评分矩阵对应的二部图执行基于点割集的图分割算法，通过移除所有的点割集，以及点割集顶点对应的边，便可以得到 k 个互不连通的连通分支，如图 2 所示。

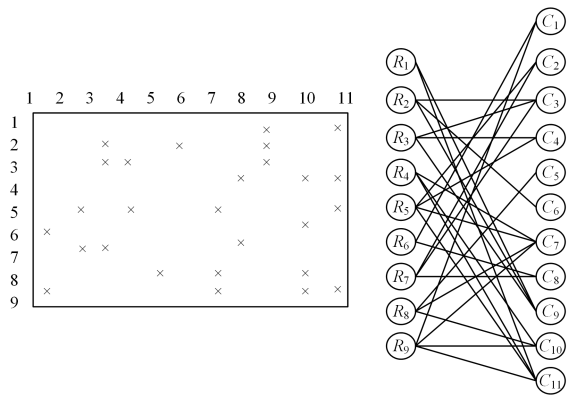


图 1 原始矩阵及其二部图

Fig. 1 Original matrix and its bipartite graph

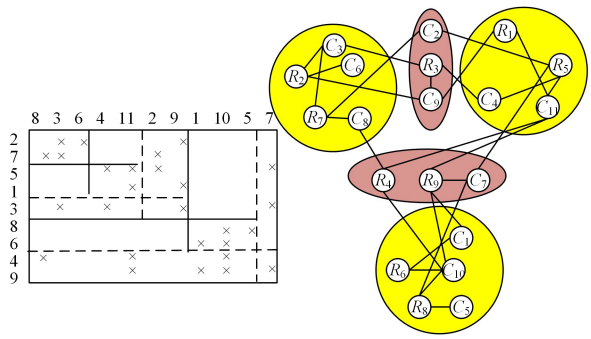


图 2 双边块对角矩阵及其二部图(电子版为彩图)

Fig. 2 Bordered block diagonal matrix and its bipartite graph

图 2 中，黄色部分为无向图的 3 个连通分支，红色部分为无向图的点割集。这 k 个连通分支对应到本文提出的双边块对角矩阵上，即 k 个对角块，点割集对应到双边块对角矩阵上的双边上。双边块对角结构可以抽象表示为：

$$X=\begin{array}{c}\begin{matrix}J_1 & J_2 & J_3\end{matrix}\\ \begin{bmatrix}D_1 & & C_1\\ & D_2 & C_2\\ R_1 & R_2 & B\end{bmatrix}\end{array}\begin{matrix}I_1\\ I_2\\ I_3\end{matrix}=\begin{array}{c}\begin{matrix}J_{11} & J_{12} & J_{B1} & J_2 & J_B\end{matrix}\\ \left[\begin{array}{ccc:c:c}D_{11} & & & & C_{11}^1\\ & D_{12} & & & C_{12}^2\\ R_{11} & R_{12} & B_1 & & C_1^3\\ \hdashline & & & D_2 & C_2\\ \hdashline R_1^1 & R_1^2 & R_1^3 & R_2 & B\end{array}\right]\end{array}\begin{matrix}I_{11}\\ I_{12}\\ I_{B1}\\ I_2\\ I_B\end{matrix} \quad (1)$$

其中，$\boldsymbol{D}$ 称为对角块，$\boldsymbol{R}$ 称为行边，$\boldsymbol{C}$ 称为列边，$\boldsymbol{B}$ 为双边交叉块，因此这个矩阵被称为双边块对角矩阵。

点割集的意义在于，能构成点割集的都是“超级用户”或者“超级物品”，对应到双边 $\boldsymbol{R}$，$\boldsymbol{C}$ 上，这些用户和物品跨社区行为十分频繁，直观地来说，就是被排列到双边的用户兴趣十分广泛，无论什么类型的物品他都会有所涉猎，超级物品在实际的系统中即受到大多数用户喜爱的热门物品。不同的连通分支表示具有相同偏好的用户以及相似属性的物品所构成的集合，通过该算法得到的对角块 $\boldsymbol{D}$，就是同一类用户和物品的集合，这就是基于二部图的社区发现图分割算法的意义所在。

进一步地，将双边块对角矩阵的双边排列与对角块排列在一起，即可得到数个子矩阵，例如将式(1)中的双边块对角矩阵排列为块对角矩阵，可以得到：

$$X=\begin{array}{c}\begin{matrix}J_{11} & J_{B1} & J_B & J_{12} & J_{B1} & J_B & J_2 & J_B\end{matrix}\\ \left[\begin{array}{ccc:ccc:c:c}D_{11} & C_{11} & C_1^1 & & & & & \\ R_{11} & B_1 & C_1^3 & & \tilde{X}_{12} & & \tilde{X}_{13} & \\ R_1^1 & R_1^3 & B & & & & & \\ \hdashline & & & D_{12} & C_{12} & C_1^2 & & \\ \tilde{X}_{21} & & & R_{12} & B_1 & C_1^3 & \tilde{X}_{23} & \\ & & & R_1^2 & R_1^3 & B & & \\ \hdashline & & & & & & D_2 & C_2\\ \tilde{X}_{31} & & & \tilde{X}_{32} & & & R_2 & B\end{array}\right]\end{array}\begin{matrix}I_{11}\\ I_{B1}\\ I_B\\ I_{12}\\ I_{B1}\\ I_B\\ I_2\\ I_B\end{matrix}$$

$$\triangleq \mathrm{diag}(\tilde{X}_1,\tilde{X}_2,\tilde{X}_3) \quad (2)$$

其中，当 $i\neq j$ 时，$X_{ij}=0$。本文基于这些对角线上的子对角块矩阵进行独立的矩阵分解，通过对重复预测的矩阵求平均，即可近似地去恢复原始矩阵。

4 基于双边块对角矩阵的矩阵分解

原始矩阵通过执行基于社区发现的点割集图分割算法，可以得到双边块对角矩阵，其中双边块对角矩阵中的每个对角块都是具有相同偏好的用户以及具有相似属性的物品的集合，如何利用这个重要的信息去进行矩阵分解，以提升预测的精度是本文需要解决的问题。

本节中，需要解决的问题如下：

(1)如何保证块对角矩阵中的子矩阵的规模是相似的。

(2)如何对每个规模相似的子矩阵进行矩阵分解。

(3)如何利用子矩阵的分解结果去预测块对角矩阵中的零块，以及恢复原始矩阵。

下文将详细阐述基于双边块对角矩阵精确恢复原始矩阵的思想，对块对角矩阵中的每个子对角块进行分解，并利用分解结果去近似原始矩阵。

4.1 基于双边块对角矩阵精确恢复原始矩阵

原始矩阵虽然通常不能直接转换为块对角矩阵，但是几乎都可以被转换为双边块对角矩阵，假设原始矩阵 $\boldsymbol{X}$ 被转换为双边块对角矩阵 $\tilde{\boldsymbol{X}}$，其中 $\tilde{\boldsymbol{X}}$ 可以被分解为：

$$\tilde{\boldsymbol{X}}\triangleq\begin{bmatrix}\boldsymbol{D}_i & \boldsymbol{C}_i\\ \boldsymbol{R}_i & \boldsymbol{B}\end{bmatrix}=\boldsymbol{U}_i\boldsymbol{V}_i^{\mathrm{T}}=\begin{bmatrix}\boldsymbol{U}_{i1}\\ \boldsymbol{U}_{i2}\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_{i1}^{\mathrm{T}} & \boldsymbol{V}_{i2}^{\mathrm{T}}\end{bmatrix} \quad (3)$$

根据式(3)，易得 $\boldsymbol{D}_i=\boldsymbol{U}_{i1}\boldsymbol{V}_{i1}^{\mathrm{T}}$，$\boldsymbol{R}_i=\boldsymbol{U}_{i2}\boldsymbol{V}_{i1}^{\mathrm{T}}$，$\boldsymbol{C}_i=\boldsymbol{U}_{i1}\boldsymbol{V}_{i2}^{\mathrm{T}}$，$\boldsymbol{B}=\boldsymbol{U}_{i2}\boldsymbol{V}_{i2}^{\mathrm{T}}$，令：

$$\boldsymbol{U}=\begin{bmatrix}\boldsymbol{U}_{11} & & & \\ & \boldsymbol{U}_{21} & & \\ & & \ddots & \\ & & & \boldsymbol{U}_{k1}\\ \boldsymbol{U}_{12} & \boldsymbol{U}_{22} & \cdots & \boldsymbol{U}_{k2}\end{bmatrix},=\begin{bmatrix}\boldsymbol{V}_{11} & & & \\ & \boldsymbol{V}_{21} & & \\ & & & \\ & & \ddots & \boldsymbol{V}_{k1}\\ \boldsymbol{V}_{12} & \boldsymbol{V}_{22} & \cdots & \boldsymbol{V}_{k2}\end{bmatrix} \quad (4)$$

可以得到：

$$\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}=\begin{bmatrix}\boldsymbol{U}_{11}\boldsymbol{V}_{11}^{\mathrm{T}} & & & & \boldsymbol{U}_{11}\boldsymbol{V}_{12}^{\mathrm{T}}\\ & \boldsymbol{U}_{21}\boldsymbol{V}_{21}^{\mathrm{T}} & & & \boldsymbol{U}_{21}\boldsymbol{V}_{22}^{\mathrm{T}}\\ & & \ddots & & \vdots\\ & & & \boldsymbol{U}_{k1}\boldsymbol{V}_{k1}^{\mathrm{T}} & \boldsymbol{U}_{k1}\boldsymbol{V}_{k2}^{\mathrm{T}}\\ \boldsymbol{U}_{12}\boldsymbol{V}_{11}^{\mathrm{T}} & \boldsymbol{U}_{22}\boldsymbol{V}_{21}^{\mathrm{T}} & \cdots & \boldsymbol{U}_{k2}\boldsymbol{V}_{k1}^{\mathrm{T}} & \sum_{i=1}^{k}\boldsymbol{U}_{i2}\boldsymbol{V}_{i2}^{\mathrm{T}}\end{bmatrix}$$

$$=\begin{bmatrix}\boldsymbol{D}_1 & & & \boldsymbol{C}_1\\ & \ddots & & \vdots\\ & & \boldsymbol{D}_k & \boldsymbol{C}_k\\ \boldsymbol{R}_1 & \cdots & \boldsymbol{R}_k & k\boldsymbol{B}\end{bmatrix} \quad (5)$$

观察分解结果发现，$\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$ 矩阵和双边块对角矩阵的唯一区别就是双边的交叉块 $\boldsymbol{B}$，$\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$ 矩阵相比双边块对角矩阵 $\tilde{\boldsymbol{X}}$，$\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$ 矩阵的双边交叉块为 $\tilde{\boldsymbol{X}}$ 矩阵的双边交叉块乘以对角块个数。根据这个思想，本文提出了基于双边块对角矩阵的矩阵分解算法，该算法的思想为将原始矩阵通过执行图分割算法排列为双边块对角矩阵后(见式(1))，再将双边块对角矩阵排列成块对角矩阵的形式(见式(2))，通过对子对角块矩阵进行矩阵分解，并对重复预测的子矩阵求平均，去恢复原始矩阵，具体算法将在 4.3 节详细描述。

4.2 得到规模相似的对角块矩阵

在实际的并行化矩阵分解中，通常一个线程对一个子矩阵进行分解，这就要求每个子矩阵的规模应该尽可能地相似，因为若存在规模相差较大的两个子矩阵，对规模较小的子矩阵进行分解的线程必定要等待规模较大的子矩阵进行分解的线程完成后才能对原始矩阵进行恢复，这会大大影响算法效率。为了得到规模尽可能相似的子对角块矩阵，本文提出算法 1，用于产生规模相似的 k 个对角块，这 k 个对角块可以对应到式(2)中，其算法描述如算法 1 所示。

算法 1 Balanced-BDF

Require：原始评分矩阵 $\mathbf{X}$，对角块矩阵目标密度 ρ，当前块对角矩阵中总对角块个数 k

Ensure：由双边块对角矩阵拼接而成的块对角矩阵 $\tilde{\mathbf{X}}$

```
1. function BDF(X,ρ,k)
2.   ρ'←ρ̄(X̃1,X̃2,X̃3,…,X̃k)
3.   if ρ'>ρ then
4.     return X̃
5.   else
6.     [D1,D2,D3,…,Dk]←SortByArea([D1,D2,D3,…,Dk])
7.     for i=1→k do
```

```
8.      [D_i^1, D_i^2] ← MetisNodeBDF(D_i)
9.      X̃_i 重新排列成 X̃_i^1, X̃_i^2
10.     if ρ̄(X̃_1, ⋯, X̃_i^1, X̃_i^2, ⋯, X̃_k) > ρ' then
11.       for j = k → i do
12.         X̃_{j++} = X̃_j
13.       end for
14.       X̃_i = X̃_i^1, X̃_{i+1} = X̃_i^2
15.       X' ← (X̃_1, X̃_2, ⋯, X̃_{k+1})
16.       Balanced-BDF(X', ρ, k+1)
17.     end if
18.   end for
19.   return X̃
20. end if
21. end function
```

基于点割集的分割算法是借助著名的开源图分割工具Metis,对双边块对角矩阵中的对角块进行分割,算法通过控制对角块的密度 ρ 来控制分割的进行,类似于式(2),如果这些对角块拼接起来的对角块子矩阵密度能得到提高,则将对角块拼接起来,并进行递归,直至所有的对角块平均密度达到预期要求后停止。

算法 1 在对对角块进行分割前,先对对角块的大小进行了一个降序的排列,在分割时也优先将面积较大的对角块先进行分割,基于面积的分割方法能保证分解出来的矩阵具有相似的面积,即子矩阵拥有相似的规模,方便后续的并行化处理。

4.3 基于对角块矩阵的矩阵分解

基于式(2)所展示的结构,根据公认的基于机器学习的矩阵分解框架[22],执行分解算法 $P=(f,D_W,C,R)$,利用每个子对角块 $\boldsymbol{X}_i$ 的分解结果 $\boldsymbol{U}_i,\boldsymbol{V}_i$ 去近似原始矩阵,分解算法每一部分的定义如下:

(1)基于矩阵分解算法中的预测函数 f,对于对角线上每个子对角块有 $\boldsymbol{X}_{ii}\approx f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})=\boldsymbol{U}_i\boldsymbol{V}_i^{\mathrm{T}}$,即矩阵分解算法中的预测函数为 $f(x)=x$。对于式(2)所示的块对角矩阵,当 $i\neq j$ 时,$X_{ij}=0$,但是可以用对角线上的分解结果对非对角线上的零矩阵块进行预测填充,即 $\boldsymbol{X}_{ij}\approx f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})=\boldsymbol{U}_i\boldsymbol{V}_j^{\mathrm{T}}$。

(2)D_W 为损失函数,$D_W(X,f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}))\geqslant 0$ 用于表示经过预测函数 $f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})$ 预测后矩阵与原始矩阵的误差,其中 $\boldsymbol{W}$ 为数据权重矩阵,是损失函数的自变量。

对于本文算法,对于整个矩阵 $\boldsymbol{X}$,令损失函数等于每个预测点的损失值的平方和与其权重的乘积,损失值用真实值与预测值之差来表示,即:

$$\begin{aligned}D_W(\boldsymbol{X},f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}))&=\sum_{i,j}D_{W_{ij}}(\boldsymbol{X}_{ij},f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})_{ij})\\&=\sum_{i,j}W_{ij}\times(\boldsymbol{X}_{ij}-f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})_{ij})^2\end{aligned}\tag{6}$$

在实际的推荐系统中,要处理的矩阵通常都是十分稀疏的矩阵,包含了大量的未观测值,为了方便处理,常置这些未观测值为 0,但并不意味着用户对它们的评分为 0。在本文提出的算法中,对于未观测值,令其权重 $\boldsymbol{W}=0$,对于本文提出的对角块矩阵中的非对角线上的对角块,有 $D_{W_{ij}}(\boldsymbol{X}_{ij},f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})_{ij})=0(i\neq j)$。

(3)C 为分解因子的约束。面向常见的含有显式评分的系统中,评分数一般都为正数,因此在本文算法中,硬性约束需要满足非负性约束,即分解结果 $\boldsymbol{U},\boldsymbol{V}$ 矩阵中不包含负数元素。

(4)R 为正则化项,为了防止过拟合问题,在本文提出的矩阵分解算法中,采用的是最常见的 ℓ_p-*norm* 正则化项:

$$\begin{aligned}R(\boldsymbol{U},\boldsymbol{V})&=\lambda_{\boldsymbol{U}}\ \|\boldsymbol{U}\|_p^p+\lambda_{\boldsymbol{V}}\|\boldsymbol{V}\|_p^p\\&=\sum_{i=1}^{k}(\lambda_{\boldsymbol{U}}\ \|\boldsymbol{U}_i\|_p^p+\lambda_{\boldsymbol{V}}\|\boldsymbol{V}_i\|_p^p)\\&=\sum_{i=1}^{k}R(\boldsymbol{U}_i,\boldsymbol{V}_i)\end{aligned}\tag{7}$$

具体算法采用的是 $p=2$ 的正则化项,即最著名的 Frobenius 正则化,采取该正则化项可以使分解结果更具有鲁棒性。

基于上述的几个参数,将矩阵分解问题转化为机器学习上的求解优化目标,对每一个子矩阵 $\boldsymbol{X}_i(1\leqslant i\leqslant k)$ 进行分解,便可以得到分解结果集合 $\boldsymbol{U}=\{\boldsymbol{U}_1,\boldsymbol{U}_2,\cdots,\boldsymbol{U}_k\},\boldsymbol{V}=\{\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_k\}$,其优化目标如下:

$$\arg\min_{(\boldsymbol{U},\boldsymbol{V})\in C}[(\boldsymbol{X}-f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}))^2+\lambda_{\boldsymbol{U}}\ \|\boldsymbol{U}\|^2+\lambda_{\boldsymbol{V}}\ \|\boldsymbol{V}\|^2)]\tag{8}$$

在每轮迭代的过程中,$\boldsymbol{U}_i,\boldsymbol{V}_j$ 的每轮迭代更新可以表示为:

$$\boldsymbol{U}_i\leftarrow\boldsymbol{U}_i+W_{ij}\times\gamma\times(D_{W_{ij}})\times\boldsymbol{X}_{ij},f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})_{ij}\times\boldsymbol{V}_j-\lambda\times\boldsymbol{U}_i\tag{9}$$

$$\boldsymbol{V}_j\leftarrow\boldsymbol{V}_j+W_{ij}\times\gamma\times(D_{W_{ij}})\times\boldsymbol{X}_{ij},f(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})_{ij}\times\boldsymbol{U}_i-\lambda\times\boldsymbol{V}_j\tag{10}$$

4.4 基于对角块矩阵的矩阵分解

本文提出的基于双边块对角矩阵的矩阵分解算法的流程总结如下:先把原始的稀疏矩阵通过迭代转换为多层双边块对角矩阵,再按照类似于式(2)的形式排列为块对角矩阵,对子对角块矩阵执行矩阵分解算法,并通过分解结果去对原始矩阵进行近似。对于原始的稀疏评分矩阵 $\boldsymbol{X}$,对它进行矩阵预测的具体实现流程为:

(1)对 $\boldsymbol{X}$ 矩阵执行基于社区发现的点割集图分割算法,将其转换为双边块对角矩阵 $\boldsymbol{X}'$。

(2)双边块对角矩阵调节为式(2)所示的块对角矩阵 $\widetilde{\boldsymbol{X}}$,$\widetilde{\boldsymbol{X}}_{I_*\sim J_*}$ 表示块对角矩阵的子矩阵。

(3)对各个子块对角矩阵 $\widetilde{\boldsymbol{X}}_{I_*\sim J_*}$ 执行矩阵分解算法 P,得到分解结果$(\boldsymbol{U}_i,\boldsymbol{V}_i)$,通过对分解结果使用预测函数 f,得到近似的分解结果 $\widetilde{\boldsymbol{X}}_i^*$,即:

$$\widetilde{\boldsymbol{X}}_i\approx f(\boldsymbol{U}_i\boldsymbol{V}_i^{\mathrm{T}})\overset{\triangle}{=}\widetilde{\boldsymbol{X}}_i^*\overset{\triangle}{=}\widetilde{\boldsymbol{X}}_{ii}^*\tag{11}$$

对角线上非零的子对角块 $\widetilde{\boldsymbol{X}}_{ii}$ 的近似已得出。

(4)根据矩阵分解算法所得到的分解结果 $\boldsymbol{U}=\{\boldsymbol{U}_1,\boldsymbol{U}_2,\cdots,\boldsymbol{U}_k\},\boldsymbol{V}=\{\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_k\}$,对块对角矩阵中非对角线上的零块进行近似。

$$\widetilde{\boldsymbol{X}}_{ij}\approx f(\boldsymbol{U}_i\boldsymbol{V}_j^{\mathrm{T}})\overset{\triangle}{=}\widetilde{\boldsymbol{X}}_{ij}^*\tag{12}$$

至此,块对角矩阵所有值的预测填充工作都已完成,整个块对角矩阵的近似已得出。

(5)对重复预测的矩阵 $\widetilde{\boldsymbol{X}}_{ij}$ 求平均,得到近似的双边块对角矩阵。以式(1)中的 R_{12} 块为例子,R_{12} 由原始双边块对角矩阵 $\boldsymbol{X}'_{I_{B1}\sim J_{12}}$ 构成,在近似原始矩阵过程中,该子块在 $\widetilde{\boldsymbol{X}}_{12},\widetilde{\boldsymbol{X}}_{22}$ 子矩阵块中共被重复预测两次(见式(2)),因此在近似原始双边块对角矩阵的 $\boldsymbol{X}^*_{I_{B1}\sim J_{12}}$ 的值为:

$$\boldsymbol{X}^*_{I_{B1}\sim J_{12}}=\frac{1}{2}(\widetilde{\boldsymbol{X}}^{*(12)}_{I_{B1}\sim J_{12}}+\widetilde{\boldsymbol{X}}^{*(22)}_{I_{B1}\sim J_{12}})\tag{13}$$

同理，对块对角矩阵中每个 $\widetilde{\boldsymbol{X}}_{I\sim J}$ 做同样的工作，便可得到近似的原始矩阵，即：

$$\boldsymbol{X}^{*}=\{\boldsymbol{X}^{*}_{I^{*}\sim J^{*}}\} \tag{14}$$

至此，矩阵分解及近似恢复原始矩阵的工作完成。

4.5 性能分析

Metis 中基于点割集的图分割算法的复杂度为 $O(n)$[23]，n 为图中边的数量，对应到评分矩阵中即非零值的个数。对于本文算法，假设最终产生 k 个对角块，通常 $k\ll n$，递归算法的递归树深度为 $O(\lg k)$，因此本文算法的复杂度为 $O(n\lg k)$。

5 实验结果

本节将本文算法与最新先进的推荐算法进行对比，将评估标准作为各种推荐算法的性能参考。实验的实验机的处理器为 AMD R3 3300X 3.8GHz CPU，实验机的内存为16GB，实验机的操作系统为 Windows 10，实现基线算法以及本文算法编程语言为 C++，编程所使用的集成开发软件为 Visual Studio 2015。

本节将先介绍实验所使用的数据集信息，然后介绍对比算法与评估标准，得出对比实验的结果后，进一步探讨影响本文提出的推荐算法精度的参数。

5.1 使用的数据集信息

对于对比实验，本文基于以下的公开的真实数据集进行研究：Movielens-1m，Book-crossing。其中，Movielens-1m 是 GroupLens 研究组对电影评分进行收集而来的数据集[1)]，是推荐系统研究中最广泛用于评估推荐系统的性能的数据集，Movielens-1m 数据集包含了 6 040 个不同的用户对约 3 900 部电影的大约 100 万个评分，评分的区间为 1～5 分。Book-crossing 是根据 bookcrossing.com 的数据编写的图书评分数据集[2)]。该数据集包含了大约 27 万个独立用户对约 27 万本书的 110 万次评分，评分区间为 1～10 分。同时，Book-crossing 是目前个性化推荐系统研究中最不密集的数据之一。两个数据集信息如表 1 所列。

表 1 实验中的数据集信息

Table 1 Statistics of two datasets

Datasets	Users	Items	Ratings	Density/%	Rating range
Movielens-1m	6040	3952	1 000 209	4.2600	1～5
Book-crossing	278 858	271 379	1 149 780	0.0014	1～10

上述两个数据集的规模、密度均有较大的差别，能很好地探究本文算法在不同场景下的表现。

5.2 算法及评价指标的选择

本文选择了以下 4 个矩阵分解算法作为基线算法，用于对比研究基于双边块对角矩阵的矩阵分解的精度。

(1)SVD[24]：奇异值分解，采用的是 Simon Funk 在 Netflix 大赛期间提出的算法，该算法采用的是 Frobenius 范数。

(2)SVD++[25]：考虑用户的历史行为对用户评分预测的影响的 SVD 算法的扩展。

(3)PMF[26]：概率矩阵分解算法，建立用户行为概率模型，并根据这个模型进行预测。

(4)NMF[27]：非负矩阵分解算法，同样采用的是 Frobenius 范数。

在对矩阵评分精度预测的研究上，本文选用的评价指标为均方根误差(RMSE)，假设测试集中共有 N 个用户 u 对物品 i 的评分 r_{ui}，若经过协同过滤算法，物品的预测值为 $\hat{r}_{ui}$，则 RMSE 可以表示为：

$$RMSE=\sqrt{\frac{\sum_{i=1}^{N}(r_{ui}-\hat{r}_{ui})^{2}}{N}} \tag{15}$$

本文采用五折交叉验证进行实验，将数据集分为 5 个部份，每次实验选择其中 4 份作为训练集，1 份作为测试集，共重复 5 次实验，每次选取的训练集都不同，通过对 5 次实验计算出来的 RMSE 值求平均，将其作为最终的 RMSE 值。

5.3 算法的分析

5.3.1 不同目标密度对对角块数量的影响

根据本文算法，生成对角块的数量是由用户设置的密度 ρ 所决定的。图 3 和图 4 给出了在两个数据集上设置的密度 ρ 和最终形成的对角块个数的关系。

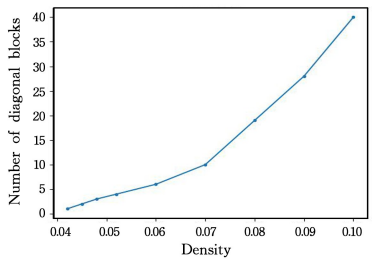


图 3 Movielens-1m 在不同密度下对角块个数

Fig. 3 Number of diagonal blocks in Movielens-1m under different densities

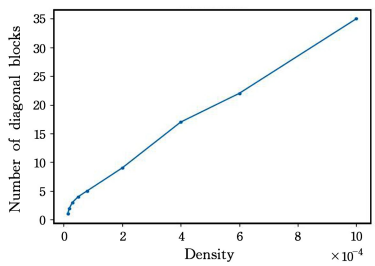


图 4 Book-crossing 在不同密度下对角块个数

Fig. 4 Number of diagonal blocks in Book-crossing under different densities

根据图 3、图 4 可以得知，随着设置的密度的增大，对角块的个数也会增多，假设设置的目标密度小于原始矩阵的密度，那么就可以将原始矩阵看作是一个对角块，生成对角块个数为 1。但是，若目标密度设置得过大，将会生成很多小规模的矩阵，这是因为随着过多对角块的生成，这些对角块的规模将会变得非常小，以 Movielens-1m 数据集生成的对角块为例，当设置的目标密度为 0.1 时，生成了 40 个平均用户个数仅为 201 个，而平均物品个数也只有 2 881 个的对角块矩阵，两个数据集在不同密度下产生的对角块中的平均用户数量以及物品数量如图 5 和图 6 所示。

1) http://grouplens.org/datasets/movielens

2) http://www2.informatik.uni-freiburg.de/~cziegler/BX/

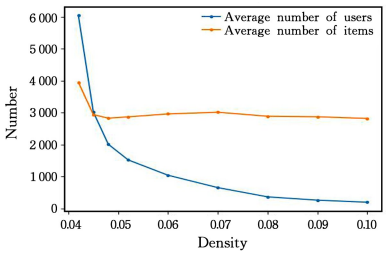


图 5 Movielens-1m 在不同密度下每个对角块平均用户数和平均物品数

Fig. 5 Average number of users and items per diagonal block in Movielens-1m under different densities

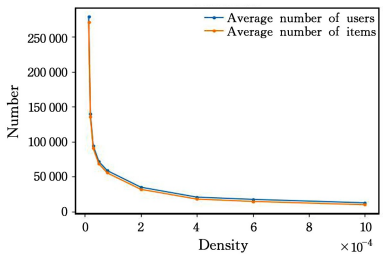


图 6 不同密度下每个对角块平均用户数和平均物品数

Fig. 6 Average number of users and items per diagonal block in Book-crossing under different densities

过多的零散小规模矩阵会十分影响矩阵分解算法对用户偏好的正确建模，后面将会设计实验验证，对角块并非越多越好。在实际的应用中，同样不需要这么多的对角块，为了并行化的处理，通常生成对角块的个数为计算机 CPU 的内核数（或整数倍），通过一条线程负责处理一个对角块即可提升矩阵分解的效率。

5.3.2 本文算法与基线算法的对比

对于矩阵分解算法 $P=(f,D_W,C,R)$ 来说，矩阵的分解因子数量 r 对评分预测的精度影响十分重大，若分解因子的数量太小，原始矩阵构建的用户偏好和物品属性模型过于简单，不能很好地去拟合原始矩阵，若分解因子数量过大，生成模型耗费的时间将会很长，并且也会带来过拟合的后果，最终的预测精度也会下降，因此选取一个恰当的分解因子数量十分重要。

将 Movielens-1m 的数据集作为实验数据，对数据集的评分矩阵执行目标密度 $\hat{\rho}=0.052$ 的双边块对角化算法，原始矩阵将产生 4 个对角块，4 个对角块的信息如表 2 所列。

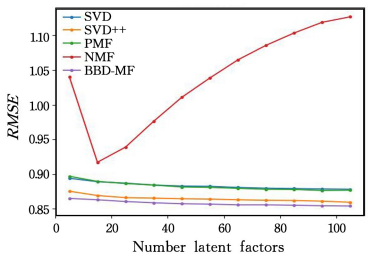
表 2 4 个对角块的信息

Table 2 Information of four diagonal blocks

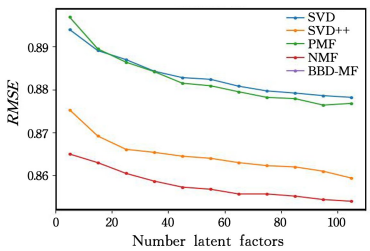
	X_1	X_2	X_3	X_4
Users	1559	1264	1413	1851
Items	2440	2766	3091	3590
Ratings	181975	163176	227915	361832
Density	0.0478	0.0467	0.0522	0.0545

为考察不同分解因子数量下，4 种基线算法以及本文提出的基于双边块对角矩阵的矩阵分解算法预测精度的区别，本文直接在原始矩阵上执行 4 种基线算法，计算并记录其 RMSE 值。作为对比，对产生的 4 个对角块并行执行本文提出的矩阵分解算法，计算并记录根据每个子对角块的分解结果去近似的矩阵的 RMSE 值。

本文将分解因子数量的范围设置为 5～105，每次实验的步长为 10。其中，SVD，SVD＋＋，NMF 算法的正则化项系数设置为 $\lambda=0.065$，PMF 算法的正则化项系数设置为 $\lambda_U=\lambda_V=0.002$，对于本文提出的矩阵分解算法，与 SVD 等算法具有相似之处，因此正则化系数也选择为 $\lambda=0.065$，五折交叉验证的实验结果如图 7 和图 8 所示。



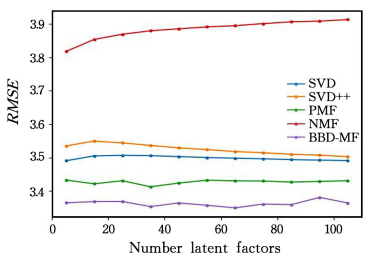
(a)5 种算法的对比



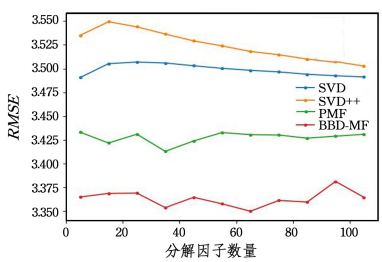
(b) 性能相近算法对比

图 7 Movielens-1m 在不同分解因子数量下 5 种算法的 RMSE

Fig. 7 RMSE of Movielens-1m for five algorithms with different numbers of lantent factors



(a)5 种算法的对比



(b) 性能相近算法对比

图 8 Book-crossing 在不同分解因子数量下 5 种算法的 RMSE

Fig. 8 RMSE of Book-crossing for five algorithms with different numbers of lantent factors

对于 Book-crossing 数据集，类似于 Movielens-1m 数据集，本文采取同样的算法，控制目标密度为 0.000 15，可以得到 8 个对角块，对这 8 个对角块采取本文算法，记录下预测的 RMSE 值，并与直接在原始矩阵上执行基线算法的结果进行比较，比较结果如图 8 所示。

在两个数据集上的实验结果如图 7、图 8 所示，5 条曲线分别为 4 种基线算法（SVD，NMF，SVD＋＋，PMF）以及本文提出的 BBD-MF 算法，实验结果表明，本文提出的算法在预测精度上均比 4 种基线算法有提升，其中在大规模稀疏数据集 Book-crossing 中表现更为突出。能得到这样的实验结果

是基于以下原因:1)算法中抽取子矩阵较原始的评分矩阵密度更高,在相对密集的子矩阵上执行协同过滤算法进行预测,能得到更高的预测结果;2)子矩阵是根据社区发现的图分割算法进行划分的,因此同一个子矩阵中的用户具有更相似的物品偏好,这对预测结果也会有提升。

同时,通过观察在 Movielens-1m 上执行 5 种算法得到的实验结果也能够发现,本文提出的基于双边块对角矩阵的矩阵分解算法在分解因子数量较小时就能得到相对较好的预测精度。这是因为对于 4 种基线算法来说,当分解因子数量较小时,模型过于简单,不足以去拟合庞大的原始矩阵,但对于本文提出的基于双边块对角矩阵的矩阵分解算法而言,算法首先得到数个对角块,矩阵分解后的预测工作是基于这些对角块的,对角块的规模相比原始矩阵要小得多,因此,即使分解因子数量较小,简单的模型也能很好地去拟合对角块矩阵。选择较小的分解因子就能获得相对较好的预测效果,这意味着能节省大量的建模时间,提高推荐算法的效率。

5.3.3 目标密度设置对预测精度的影响

本节将讨论目标密度设置对预测精度的影响,通过前面的实验已知在合理设置目标密度的前提下,对角块个数总是与目标密度成正比的,但是在实际的系统中,往往不需要过多的子矩阵。本节实验对 Movielens-1m 以及 Book-crossing 数据集中的评分矩阵设置不同的目标密度,对其执行双边块对角化算法,并产生对角块。根据 5.3.2 节的实验可知,在 Movielens-1m 数据集上,当分解因子数量为 65 时,预测的精度就趋于平稳,预测的精度也可以被接受;在 Book-crossing 数据集上分解因子个数为 35 时,预测的精度相对较好。因此,对于 Movielens-1m 数据集,固定分解因子数量为 65,对 Book-crossing 数据集固定分解因子数量为 35,并对这两个数据集产生的对角块执行文中提出的矩阵分解算法,正则化系数的设置与 5.3.2 节相同,其结果如图 9 和图 10 所示。

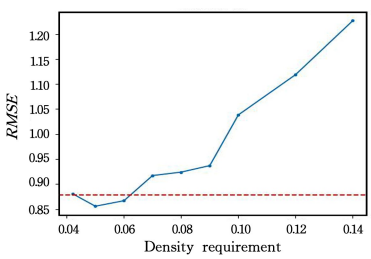


图 9 Movielens-1m 数据集在不同对角块密度下 BBD-MF 算法的 RMSE

Fig. 9 RMSE of BBD-MF algorithm for Movielens-1m on different densities of the diagonal block

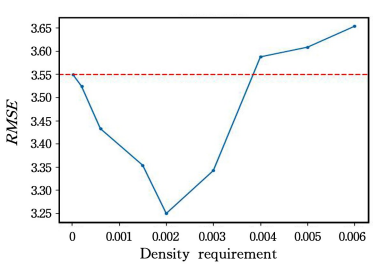


图 10 Book-crossing 数据集在不同对角块密度下 BBD-MF 算法的 RMSE

Fig. 10 RMSE of BBD-MF algorithm for Book-crossing on different densities of the diagonal block

图 9、图 10 中初始点为评分矩阵的原始密度下的 RMSE,即不对矩阵进行分割,执行本文提出的矩阵分解算法,并进行预测的 RMSE。为了直观比较,根据初始点的值绘制虚线,虚线表示不对矩阵进行分割直接进行矩阵分解的均方根误差。从图 9、图 10 中可以得知,RMSE 一开始随着目标密度的提升而降低,但是当目标密度设置得过大时,随着对角块的增多,RMSE 反而会比初始矩阵更大,矩阵的规模过小,将会大大影响推荐算法对用户偏好以及物品属性的建模。上述实验结果表明,对角块并不是越多越好,在实际的应用中,应选择合适的密度,产生合适数量的对角块进行预测。

5.3.4 并行化效率分析

在此部分,将对本文算法的并行化效率进行分析,实验的实验机核心数为 4 核 8 线程的 CPU,因此本次实验通过控制输入的密度,将原始矩阵分解为 8 个对角块,在 4 个物理内核上运行 8 个线程,每个线程对一个对角块进行矩阵分解算法,并记录形成对角块矩阵的时间与通过分解结果来预测原始矩阵所用的时间之和。对于基线算法,直接对原始矩阵执行 4 种基线算法,记录每种算法分解矩阵并完成预测所需的时间。算法的分解因子个数以及正则化项参数均与 5.3.3 节的实验保持一致,算法执行完成后,结果如表 3 所列。

表 3 在两个数据集下算法耗时的对比

Table 3 Computational time of all algorithm on different dataset

Algorithm	Movielens-1m	Book-crossing
SVD	17.6 s	23.4 s
NMF	43.3 s	76.5 s
SVD++	27.4 min	1.05 h
PMF	18.42 s	22.5 s
BBD-MF	12.5 s	20.7 s

本文算法通过将原始矩阵分解为若干个规模相似的对角块,每个对角块都能独立执行矩阵分解算法,这使得矩阵分解算法的并行化效率大幅度提升。同时,当整体的数据发生变化后,不需要对整个矩阵进行重复的分解,只需要对发生改变的子矩阵进行重新分解即可,通过重新分解得来的分解结果去预测原始矩阵。并且对于越稀疏的矩阵,本文算法对比基线算法的提升越大,通过将原始矩阵分解成若干个对角块,使得不需要用很复杂的分解方法去训练模型,模型也会有较好的效果,这对实际系统中处理大规模稀疏矩阵有着重大的意义。

结束语 为了缓解推荐系统中数据稀疏性带来的种种问题,本文受社区发现算法启发,考虑到同一社区中的用户具有相似的兴趣偏好,本文从矩阵内部结构出发,使用基于社区发现的点割集图分割算法,对原始矩阵进行划分,可得到双边块对角矩阵,通过将双边块对角矩阵的双边和对角块拼接,能得到密度比原始矩阵更高并且可以并行进行分解的子矩阵。

本文实验部分,基于 Movielens-1m 数据集以及大规模稀疏数据集 Book-crossing 进行了对比实验。实验结果表明,通过设置合理的密度参数,就能生成合适数量的子对角块矩阵,通过合理设置正则化项、分解因子数量等参数,对比基线算法,本文算法能缓解数据稀疏性带来的问题,可以得到更好的预测精度,同时算法耗时也会更短,特别是在大规模、极其稀疏的数据集下表现更优。

本文在缓解推荐系统冷启动问题上还有非常大的提升空间,这有待更进一步开展实验进行详细的研究。同时,在未来

的研究中,拟将本文算法融入分布式大数据框架之中,以验证本文算法在处理特大规模、极其稀疏数据时的优越性。

参 考 文 献

[1] BOBADILLA J,ALONSO S,HERNANDO A. Deep learning architecture for collaborative filtering recommender systems[J]. Applied Sciences,2020,10(7):2441.

[2] WU L,GE Y,LIU Q,et al. Modeling the Evolution of Users' Preferences and Social Links in Social Networking Services[J]. IEEE Transactions on Knowledge and Data Engineering,2017,29(6):1240-1253.

[3] BOBADILLA J,ORTEGA F,GUTIERREZ A,et al. Classification-based Deep Neural Network Architecture for Collaborative Filtering Recommender Systems[J]. International Journal of Interactive Multimedia & Artificial Intelligence, 2020,6(1):68-77.

[4] FENG C,LIANG J,SONG P,et al. A fusion collaborative filtering method for sparse data in recommender systems[J]. Information Sciences,2020,521:365-379.

[5] LI D S,CHEN C,LV Q,et al. An algorithm for efficient privacy-preserving item-based collaborative filtering[J]. Future Generation Computer Systems,2016,55:311-320.

[6] SAMMEL M D,RYAN L M. Latent variable models with fixed effects[J]. Biometrics,1996,52:650-663.

[7] DEERWESTER S,DUMAIS S T,FURNAS G W,et al. Indexing by latent semantic analysis[J]. Journal of the American society for information science,1990,41(6):391-407.

[8] LEE D D,SEUNG H S. Learning the parts of objects by non-negative matrix factorization[J]. Nature,1999,401(6755):788-791.

[9] MNIH A,SALAKHUTDINOV R R. Probabilistic matrix factorization[J]. Advances in Neural Information Processing Systems,2007,20:1257-1264.

[10] NAJAFABADI M K,MAHRIN M N,CHUPRAT S,et al. Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data [J]. Computers in Human Behavior,2017,67(FEB.):113-128.

[11] WANG Y G,SONG Z Z,XIAO C L. Collaborative filtering recommendation algorithm based on improved clustering and matrix factorization[J]. Journal of Computer Applications,2018,38(4):1001-1006.

[12] ABBASI-MOUD Z,VAHDAT-NEJAD H,SADRI J. Tourism recommendation system based on semantic clustering and sentiment analysis[J]. Expert Systems with Applications,2021,167:114324.

[13] AALDERING L J,LEKER J,SONG C H. Recommending untapped M&A opportunities:A combined approach using principal component analysis and collaborative filtering[J]. Expert Systems with Applications,2019,125:221-232.

[14] NGUYEN T T. On the Edge and Cloud:Recommendation Systems with Distributed Machine Learning[C]// 2021 International Conference on Information Technology(ICIT). IEEE,2021:929-934.

[15] YU H F,HSIEH C J,SI S,et al. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems[C]// 2012 IEEE 12th International Conference on Data Mining. IEEE Computer Society,2012:765-774.

[16] LIU Q Q,LUO Y L,WANG Y F,et al. Hybrid Recommendation Algorithm Based on SVD Filling[J]. Computer Science,2019,46(S1):468-472.

[17] SU Y,ZHOU K,ZHANG X,et al. A parallel multi-objective evolutionary algorithm for community detection in large-scale complex networks[J]. Information Sciences,2021,576:374-392.

[18] OSABA E,DEL SER J,CAMACHO D,et al. Community detection in networks using bio-inspired optimization:Latest developments,new results and perspectives with a selection of recent meta-heuristics[J/OL]. Applied Soft Computing, 2020, 87. https://doi.org/10.1016/j.asoc.2019.106010.

[19] FORTUNATO S,HRIC D. Community detection in networks:A user guide[J]. Physics reports,2016,659:1-44.

[20] ZHAO W J,ZHANG F B,LIU J L. Review on community Detection in Complex Networks[J]. Computer Science, 2020,47(2):10-20.

[21] GLIGORIJEVIĆ V,PANAGAKIS Y,ZAFEIRIOU S. Non-negative matrix factorizations for multiplex network analysis[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2018,41(4):928-940.

[22] SINGH A P,GORDON G J. Relational learning via collective matrix factorization[C]// Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery. 2008:650-658.

[23] KARYPIS G,KUMAR V. A fast and high quality multilevel scheme for partitioning irregular graphs[J]. SIAM Journal on Scientific Computing,1998,20(1):359-392.

[24] KOREN Y,BELL R,VOLINSKY C. Matrix factorization techniques for recommender systems[J]. Computer,2009,42(8):30-37.

[25] KOREN Y. Factorization meets the neighborhood:a multifaceted collaborative filtering model[C]// Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery. 2008:426-434.

[26] SALAKHUTDINOV R,MNIH A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo[C]// Proceedings of the 25th International Conference on Machine Learning. Association for Computing Machinery. 2008:880-887.

[27] LUO X,ZHOU M C,XIA Y B,et al. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems[J]. IEEE Transactions on Industrial Informatics,2014,10(2):1273-1284.

HE Yi-chen, born in 1998, postgraduate, is a student member of China Computer Federation. His main research interests include recommendation system and data mining.

GU Wan-rong, born in 1982, Ph.D, assistant professor. His main research interests include recommendation system, information retrieval and big data processing.