文章编号：1003-0077(2017)04-0174-10

SCMF：一种融合多源数据的软约束矩阵分解推荐算法

满 彤，沈华伟，黄俊铭，程学旗

（中国科学院计算技术研究所 中国科学院网络数据科学与技术重点实验室，北京 100190）

摘 要：数据稀疏是推荐系统面临的主要挑战之一。近年来，多源数据融合为解决数据稀疏问题提供了新思路。然而，现有方法大多假设对象在不同数据源中具有相同的表示，这种硬约束方式无法刻画对象在不同数据源中的差异性。该文提出一种基于软约束矩阵分解的推荐算法，通过约束不同数据源中对象的隐因子向量，能够同时刻画同一对象表示的共性及其在不同数据源中的差异性。在两个数据集上的实验表明，该文提出的软约束矩阵分解算法在准确率方面优于现有的单数据源推荐算法和多源数据硬约束融合推荐算法，可以有效解决推荐系统面临的数据稀疏问题。

关键词：协同过滤；推荐系统

中图分类号：TP391　　**文献标识码**：A

SCMF: A Matrix Factorization Model With Soft Constraint for Multi-Source Recommendation

MAN Tong, SHEN Huawei, HUANG Junming, CHENG Xueqi

(CAS Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Data sparsity is a challenge forrecommender systems. In recent years, the integration of data from different sources provides a promising direction for the solution of this issue. However, most existing methods for data integration assume that the representation of a single user/item is the same across different contexts, which blocksthe depiction of the distinct characteristics of different contexts. In this paper, we propose a matrix factorization model with soft constraint that the difference between the representations of a single user/item is minimized together with the error function of matrix factorization model. Experiments on two datasets demonstrate that the proposed model outperforms thestate-of-the-art models, especially on the case where the data is sparse in only one resource.

Key words: collaborative filtering; recommender system

1 引言

互联网上规模快速增长的数据带来了严峻的信息过载问题，推荐系统在此背景下应运而生[1]。通过分析用户的历史行为数据，推荐系统能够根据用户兴趣向用户推荐其感兴趣的对象，例如电影、书籍、商品等。协同过滤算法是推荐系统中常用的一种算法。目前的协同过滤算法可以大致分为两类，第一类是基于邻居的算法[2-3]，第二类是基于模型的算法[3-4]。其中，矩阵分解[5-6]是一种主流的推荐算法。

矩阵分解算法将协同过滤问题抽象为一个矩阵填充问题。用户的历史数据被抽象成一个打分矩阵，矩阵分解将打分矩阵分解为一个用户因子矩阵和一个物品因子矩阵。用户未观察到的打分数据，可以通过用户因子矩阵和物品因子矩阵预测出来。矩阵分解算法的精度依赖于打分矩阵。在实际应用中，打分矩阵往往比较稀疏，给推荐系统带来了稀疏性问题。推荐系统的用户端和物品端都存在着稀疏

收稿日期：2016-05-05　**定稿日期**：2016-06-02

基金项目：国家自然科学基金(61202215,61232010,61425016)；信息网络安全公安部重点实验室开放课题

性问题。例如，Amazon[①] 上购买家具的用户可能在未来几年内不会再产生家具购买行为；一个刚刚进入豆瓣[②]的新用户可能只会给很少的物品打分。另外在物品端，大部分用户的兴趣都集中在少数热门物品上，形成长尾效应[2]。

推荐系统的发展过程中，存在大量的研究工作旨在解决稀疏性问题。现有的方法可以大致分为两类。第一类方法在模型层面，通过改进推荐模型、优化算法来提高推荐性能。这类方法虽然的确能够带来一定的性能提升，但是并没有从根本上解决数据的稀疏性问题。在实际场景中，用户和物品都不是孤立存在的。一个在新浪微博[③]中有历史信息的用户可能也会同时出现在校内网[④]中；一个出现在时光网[⑤]中的电影，有很大的可能也存在于豆瓣网中。基于此，近年来解决数据稀疏性问题的研究工作开始转向第二类方法，即通过融合多个不同的数据源中的信息，来解决推荐算法面临的数据稀疏问题。例如，豆瓣网中的同一个用户，可能同时会对电影、音乐、书籍三类物品打分，在用户和物品之间形成三个不同类型的打分矩阵，用户同时出现在这三个矩阵中；类似地，一部电影在豆瓣网上得到豆瓣用户的一系列评分，同时会得到时光网用户的一系列评分，豆瓣网和时光网就扮演着不同的数据源，形成用户和电影之间的两个打分矩阵。

我们从两个电影评分数据集（Netflix[⑥] 和 MovieLens[⑦]）中随机选取了 140 部电影，图 1 中展示了这些电影在两个数据集合中的分布。从图中可以看到，整体打分数分布呈线性相关，在一个场景中打分数很多的电影有很大的可能在另一个场景中也有很多的打分。然而也有不少的电影在两个场景中存在着不同的稀疏程度。例如，法国动作电影 *Le Professionnel* 在 Netflix 中仅仅有少数的打分，但是在 MovieLens 中获得了较多的反馈信息；西部探险电影 *Last of the Dogmen* 在 MovieLens 中被很多用户关注，而在 Netflix 中较为冷淡。

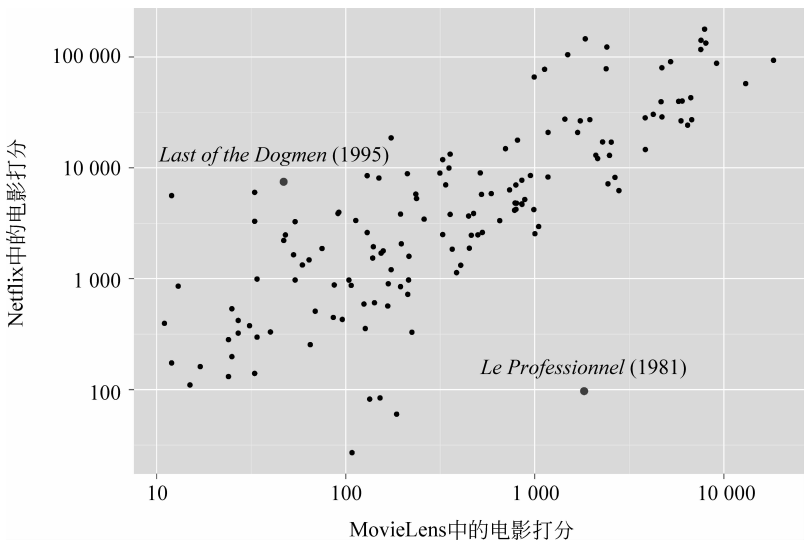

图 1　两个电影评分数据集中电影评分数分布。

多源数据融合的推荐算法，其基本思想是将从一个数据源的打分矩阵中学习到的有关用户和物品的知识，应用到另外一个数据源中，综合利用多个数据源的知识解决数据稀疏问题，从而来提高推荐算法的准确率。特别是，一些用户和物品在某些数据源中的评分数目较少，而在另外一些数据源中的评分数目较多，通过整合这些不同数据源的打分信息，建立一个多数据源融合的模型，使各个数据源的信息彼此补充，解决数据稀疏问题。

本文基于推荐系统中广泛使用的矩阵分解算法，通过对不同数据源中的用户/物品的表示向量进行约束，提出了一种基于软约束的矩阵分解推荐算法。我们通过引入约束项，来约束同一个用户（或者物品）出现在不同数据源中的因子的相似性。在 Netflix 和 MovieLens 数据集上的实验表明，本文提

① http：//www. amazon. com
② http：//www. douban. com
③ http：//www. weibo. com
④ http：//www. renren. com
⑤ http：//www. mtime. com
⑥ http：//www. netflix. com
⑦ http：//www. movielens. com

出的软约束矩阵分解算法在准确率方面优于现有的单数据源推荐算法和多源数据硬约束融合推荐算法，可以有效解决推荐系统面临的数据稀疏问题。

本文的组织结构如下：第二节介绍相关工作；第三节介绍基本的矩阵分解模型、SCMF模型及学习算法；第四节给出我们模型的推断算法；实验结果与分析在第五节给出，最后一节总结我们的工作以及对未来的工作进行展望。

2 相关工作

协同过滤(collaborative filtering)算法是推荐系统中使用广泛且有效的算法[1]。协同过滤算法主要可以分为两种：一种是基于邻居(neighborhood-based)的算法，一种是基于模型(model-based)的算法。基于邻居的算法[2]主要通过寻找相似的用户或者物品来完成推荐。基于模型的算法通过利用用户在物品上的历史信息学习出一个模型，进而利用模型来预测用户未来可能会喜欢的物品。矩阵分解模型[4-5](matrix factorization)是近年来非常流行的一种基于模型的推荐算法。矩阵分解模型通过分解用户和物品间的打分矩阵，利用得到的用户隐因子矩阵和物品隐因子矩阵来预测缺失的分数。矩阵分解模型假设一个用户对一个物品的打分是由该用户和物品的隐因子向量相互作用得到的，其中最常用的相互作用假设就是向量点积。Ruslan[6]对矩阵分解做了很好的概率化的解释，进一步提高了矩阵分解推荐算法的准确率。

推荐系统面临的一个挑战是数据稀疏问题[7]。用户的打分集中在少数物品上，同时少数用户给出了大部分的打分，形成长尾效应[8]。为了解决数据稀疏问题，Ma[9]和Liu等人引入用户之间的社交关系来指导模型的学习过程，Noam[10]等人考虑引入对象间的层次关系来对模型中的参数进行控制。Chen[11]等人提出了SVDFeature算法，一个基于特征的矩阵分解框架。该框架能够融合各种类型的信息，例如时序信息、邻居信息、物品结构信息等。Yu[12]等人将多种类型的信息整合起来，构建成一个异质信息网络。通过使用元路径的方式，构建出多个偏好矩阵，分别应用推荐算法，最后的推荐结果是在所有偏好矩阵的推荐结果上的一个整合。这些方式都是通过引入一些打分矩阵之外的信息，来缓解数据稀疏问题。

然而，上述工作均在用户和物品间的单个数据源上进行，真实情况下用户和物品之间可以形成多个打分矩阵，特别是当数据来自多个数据源时。近年来，在推荐算法方面，涌现出了大量的基于多源数据矩阵的研究工作[13-15]。Berk[16]等人提出了基于邻居的多源的协同过滤算法，通过传递不同数据源中用户和物品的相似性来缓解单个数据源中用户或对象的打分稀疏问题。Pan[17]等人提出了一个CTS(coordinate system transfer)模型，基于迁移学习的想法，通过在一个较为稠密的用户物品评分矩阵投射到一个子空间坐标系，然后将该坐标系作为信息迁移到另一个稀疏矩阵中，从而缓解稀疏性问题。Singh[18]等人提出了协同矩阵分解模型(CMF)，CMF同时分解多个数据源的打分矩阵，当同一个对象(用户或物品)出现在多个数据源中时，该对象在所有数据源中的隐因子向量都是一致的。CMF模型考虑到了相同对象在不同的数据源中的同质性，而忽略了其异质性。例如，受到用户群体，网站广告策略的影响，一部电影在两个电影评分网站中的行为会存在着差异性；同样，同一个用户在不同的数据源中的行为也会有一些差异性，在一个网站中很活跃的用户，在另一个网站中可能只是一个很少发布信息的观看者。近年来，一些多源数据融合的推荐算法提出，用于解决跨场景推荐的问题[19-20]。

3 基于软约束矩阵分解的多源数据推荐算法

如之前所述，推荐系统旨在预测用户对未知物品的偏好程度，可以形式化为矩阵填充问题。用户的历史数据以一个 $M\times N$ 的偏好打分矩阵 $\boldsymbol{R}$ 来表示，其中 M 是用户的个数，N 是物品的数量，用户 i 对物品 j 的打分由 R_{ij} 表示。在真实场景中，每个用户往往只会对一部分的物品进行评分，因此 $\boldsymbol{R}$ 通常是稀疏的。我们通过对打分矩阵 $\boldsymbol{R}$ 的部分观测，来推断 $\boldsymbol{R}$ 的全部单元的值。

矩阵分解模型中，模型假设用户对一个物品的打分是由用户的因子向量和物品的因子向量共同作用得到。模型定义用户的隐因子向量矩阵为 $\boldsymbol{U}\in\Re^{K\times M}$，其中第 i 列 $\boldsymbol{U}_i$ 表示用户 i 在隐空间中的因子向量，K 为用户隐因子向量的维度。在物品端，模型定义 $\boldsymbol{V}\in\Re^{K\times N}$ 表示物品的隐因子矩阵。

矩阵分解模型中，用户 i 对物品 j 的打分 R_{ij} 建模为以 $\boldsymbol{U}_i^T\boldsymbol{V}_j$ 为均值的高斯分布，其条件概率表示如式(1)。

$$p(R_{ij}\mid \boldsymbol{U},\boldsymbol{V},\sigma^2)=\mathrm{N}(R_{ij}\mid \boldsymbol{U}_i^T\boldsymbol{V}_j,\sigma^2) \tag{1}$$

其中 $\mathrm{N}(x\mid\mu,\sigma^2)$ 是以 μ 为均值、σ^2 为方差的高斯分布的概率密度函数。

同时，用户的隐因子向量 $\boldsymbol{U}_i$ 和物品的隐因子向量 $\boldsymbol{V}_j$ 被建模为均值为 $\vec{0}$ 的高斯分布。

$$p(\boldsymbol{U}_i\mid\sigma_u^2)=\mathrm{N}(\boldsymbol{U}_i\mid\vec{0},\sigma_u^2) \tag{2}$$

$$p(\boldsymbol{V}_j\mid\sigma_v^2)=\mathrm{N}(\boldsymbol{V}_j\mid\vec{0},\sigma_v^2) \tag{3}$$

矩阵分解的目标是最大化观察到打分矩阵的概率，既最大化如下的函数。

$$p(\boldsymbol{R}\mid\boldsymbol{U},\boldsymbol{V},\sigma^2)=\prod_{i=1}^{M}\prod_{j-1}^{N}[\mathrm{N}(R_{ij}\mid\boldsymbol{U}_i^T\boldsymbol{V}_j,\sigma^2)]^{I_{ij}} \tag{4}$$

其中，I_{ij} 是一个指示函数，当用户 i 对物品 j 有打分信息时 I_{ij} 的值为 1，反之为 0。将目标函数展开，优化问题可以转化为如下对因子矩阵的推断问题[6]。

$$[\boldsymbol{U},\boldsymbol{V}]=\underset{U,V}{\operatorname{argmin}}\frac{1}{2}\sum_{i=1}^{M}\sum_{j-1}^{N}I_{ij}\,(R_{ij}-\boldsymbol{U}_i^T\boldsymbol{V}_j)^2+\frac{\lambda_u}{2}\parallel\boldsymbol{U}\parallel_F^2+\frac{\lambda_v}{2}\parallel\boldsymbol{V}\parallel_F^2 \tag{5}$$

其中 $\lambda_u,\lambda_v>0$，可以看作是对于用户因子矩阵和物品因子矩阵的正则约束系数。$\parallel*\parallel_F$ 表示 Frobenius 范数。根据推断的 U 和 V 的值，对于一个未知的用户物品对 (k,l)，可以预测出打分的期望值为 $\hat{R}_{kl}=U_k^TV_l$。

矩阵分解模型工作在单个数据源的场景中。如果某些用户或物品同时出现在多个数据源中，这些重叠用户或物品在各个数据源中的隐因子向量应该具有一定的相关性。如何刻画这种相关性就是多数据源推荐系统的出发点。通过整合多个数据源里的用户和物品的信息，我们可以改进对用户因子和物品因子估计的准确性。我们以物品重叠的多数据源为例展开本文接下来的讨论，根据对称性，这一讨论可以应用于用户重叠的场景。例如，一个物品在某一数据源中的可用数据非常稀疏，难以准确地估计它的隐因子向量，可以用它在另一数据源中的向量辅助估计。为了整合不同数据源中的数据，协同矩阵分解模型(CMF)约束不同数据源中的同一物品的隐因子保持一致。考虑具有共同物品的两个数据源，问题可以形式化为：

$$\begin{aligned}[\boldsymbol{U}^{(1)},\boldsymbol{U}^{(2)},\boldsymbol{V}]=&\underset{\boldsymbol{U}^{(1)},\boldsymbol{U}^{(2)},\boldsymbol{V}^{(1)},\boldsymbol{V}^{(2)}}{\operatorname{argmin}}\frac{1}{2}\sum_{i=1}^{M_1}\sum_{j=1}^{N}I_{ij}^{(1)}\,(R_{ij}^{(1)}-\boldsymbol{U}_i^{(1)\mathrm{T}}\boldsymbol{V}_j)^2+\frac{\lambda_u}{2}\parallel\boldsymbol{U}^{(1)}\parallel_F^2\\&+\frac{1}{2}\sum_{i=1}^{M_2}\sum_{j=1}^{N}I_{ij}^{(2)}\,(R_{ij}^{(2)}-\boldsymbol{U}_i^{(2)\mathrm{T}}\boldsymbol{V}_j)^2+\frac{\lambda_u}{2}\parallel\boldsymbol{U}^{(2)}\parallel_F^2+\frac{\lambda_v}{2}\parallel\boldsymbol{V}\parallel_F^2\end{aligned} \tag{6}$$

其中，$\boldsymbol{R}^{(1)}$ 与 $\boldsymbol{R}^{(2)}$ 分别表示两个数据源中的打分矩阵，$\boldsymbol{U}^{(1)}$ 与 $\boldsymbol{U}^{(2)}$ 分别表示两个数据源中用户的隐因子向量矩阵，M_1 和 M_2 分别表示第一个和第二个数据源中用户的数量，N 表示物品的数量。CMF 基于隐因子向量共享的机制，同时分解两个打分矩阵。然而，这一机制仅仅考虑建模了同一对象在不同源中的相似性，没有考虑到差异性。在实际中，用户在不同系统中的表现可能会存在着差异性，例如一个用户在社交网站上和音乐网站上可能表现出不同的兴趣分布；同一个物品在不同的数据源中，受到环境的影响，可能呈现出不同的打分分布。

基于以上讨论，本文中我们提出了一种基于软约束的协作矩阵分解模型(soft-constraint matrix factorization model，SCMF)。我们在模型中，同时考虑到了出现在多个数据源中的同一个对象的隐因子向量的相似性和差异性。对于同一个对象，我们在每个数据源中都为该对象定义一个局部的隐因子向量，用以建模其在不同数据源中的差异性；同时，我们在整体的目标函数中，添加一个相似函数作为约束，限制同一对象在多个数据源中的隐因子的相似性。

同样以物品重叠的多数据源场景为例，我们的方法可以抽象为如下的目标函数。

$$\begin{aligned}[\boldsymbol{U}^{(1)},\boldsymbol{U}^{(2)},\boldsymbol{V}^{(1)},\boldsymbol{V}^{(2)}]=&\underset{\boldsymbol{U}^{(1)},\boldsymbol{U}^{(2)},\boldsymbol{V}^{(1)},\boldsymbol{V}^{(2)}}{\operatorname{argmin}}\frac{1}{2}\sum_{i=1}^{M_1}\sum_{j=1}^{N}I_{ij}^{(1)}\,(R_{ij}^{(1)}-\boldsymbol{U}_i^{(1)\mathrm{T}}\boldsymbol{V}_j^{(1)})^2+\frac{\lambda_u}{2}\parallel\boldsymbol{U}^{(1)}\parallel_F^2+\frac{\lambda_v}{2}\parallel\boldsymbol{V}^{(1)}\parallel_F^2\\&+\frac{1}{2}\sum_{i=1}^{M_2}\sum_{j=1}^{N}I_{ij}^{(2)}\,(R_{ij}^{(2)}-\boldsymbol{U}_i^{(2)\mathrm{T}}\boldsymbol{V}_j^{(2)})^2+\frac{\lambda_u}{2}\parallel\boldsymbol{U}^{(2)}\parallel_F^2+\frac{\lambda_v}{2}\parallel\boldsymbol{V}^{(2)}\parallel_F^2\\&+\alpha\sum_{i=1}^{N}f_{sim}(\boldsymbol{V}_i^{(1)},\boldsymbol{V}_i^{(2)})\end{aligned} \tag{7}$$

其中 $\boldsymbol{U}^{(1)}$ 与 $\boldsymbol{U}^{(2)}$ 分别表示两个数据源中用户的隐因子向量矩阵，$\boldsymbol{V}^{(1)}$ 与 $\boldsymbol{V}^{(2)}$ 分别表示两个数据源中物品的隐因子向量矩阵。目标函数中的前两项和传统的矩阵分解算法一致，希望针对每个源数据的打分矩阵分解出来的因子矩阵能够较好地拟合真实的情况；同时我们引入相似函数约束项，$f_{\text{sim}}(*,*)$ 函数，用来度量两个向量之间的相似程度，控制出现在多个数据源中的用户或者物品的隐因子向量在不同的数据源中依然有一定的相似性。其中参数 α 控制对相似性的约束程度，其值越大对相似的约束越大。当 α 取 0 时，我们的方法与矩阵分解算法一致；当 α 取 $+\infty$ 时，相当于约束所有用户和物品在所有的数据源中的隐因子向量都一致，我们的模型与 CMF 模型一致。

$f_{\text{sim}}(*,*)$ 函数是模型里非常重要的一部分。在本文中，我们考虑了两种类似的相似函数，第一种是线性的约束函数，第二种是非线性的约束函数。

在线性约束函数的情形下，我们通过一个距离函数 $\text{dist}(*,*)$ 来考虑隐因子向量之间的相似性。我们需要从两个方面考虑距离函数的选择。第一，距离函数需要较好地刻画两个隐因子向量之间的差异性；第二，距离函数的形式需要利于优化求解。基于这两点考虑，我们选用了如下的负点积距离函数。

$$f_{\text{sim}}(\boldsymbol{V}_i^{(1)},\boldsymbol{V}_i^{(2)}) = \text{dist}(\boldsymbol{V}_i^{(1)},\boldsymbol{V}_i^{(2)})$$
$$= -\boldsymbol{V}_i^{(1)} \cdot \boldsymbol{V}_i^{(2)} = -\sum_{k=1}^{K} V_{ik}^{(1)} V_{ik}^{(2)} \quad (8)$$

可以看到，我们选取点积的负值作为距离函数。两个向量的点积可以当做向量相似程度的一个度量，因此将点积取负值后满足我们之前提出的一个条件；第二，点积的形式非常利于优化。使用线性约束形式的 SCMF 模型被记为 SCMF(Lin)。

隐因子模型的一个特点是，隐因子向量的每个维度代表的含义并不是传统意义上的话题的含义。给定两个打分矩阵，各自进行矩阵分解之后，每个维度代表的含义可能是不一样的。因此线性模型这种直接对比各个维度的方式可能会错误地约束了隐因子向量。出于这一点考虑，我们另外采用了一种非线性的方式来约束隐因子向量。具体的我们采用多层感知机(multi-layer perceptron，MLP)的方式来约束不同空间里的隐因子向量。

采用多层感知机约束函数的 SCMF 模型记为 SCMF(MLP)，如图 2 所示。

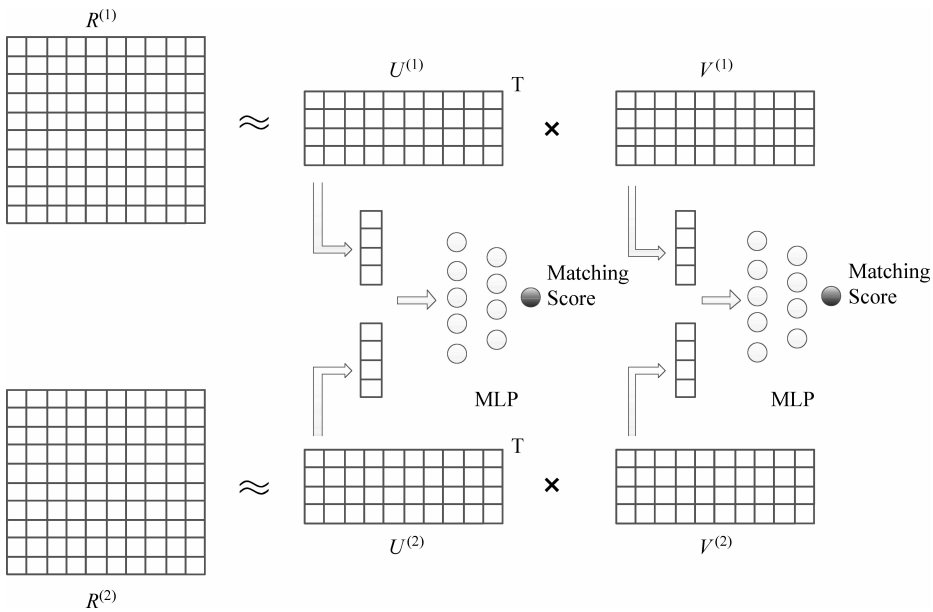

图 2 软矩阵约束矩阵分解(MLP)示例图

我们标记多层感知机约束函数为 $f_{\text{sim}}(*,*;\phi)$。其中 ϕ 是函数的参数，为矩阵形式。约束函数的输入为两个向量，输出为 0 到 1 之间的匹配分数。对于两个打分矩阵中属于一个对象的隐因子向量，我们期望约束函数输出一个较高的分数。如图 3 所示，我们考虑一个单一隐层的感知机，输入为两个因子向量拼接起来的向量 $\boldsymbol{x}$，通过两层的非线性变换，最终得到一个输出的分数 z。我们选择激活函数 $h(*)$ 为 Sigmoid 函数 $h(*)=1/(1+e^{-*})$，模型的参数为 $\phi=[W_1,W_2,b_1,b_2]$，因此对于 SCMF(MLP)模型的参数分为两部分，第一部分是用户和物品的隐因子向量，第二部分是相似函数的参数。

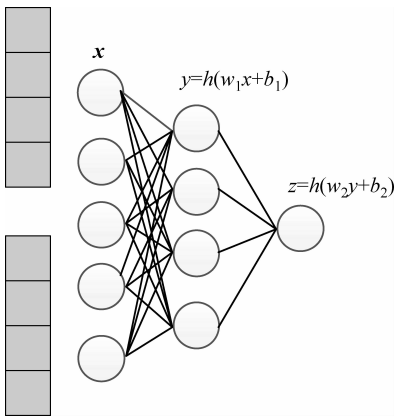

图 3　MLP 函数示意图

此外，考虑到用户和物品都可能出现在不同的源中，我们的模型可以推广为以下形式，其中源数据矩阵的个数为 L，用户个数为 M，物品的个数为 N。

$$\begin{aligned}[\boldsymbol{U}^{(1)},\cdots,\boldsymbol{U}^{(L)},\boldsymbol{V}^{(1)},\cdots,\boldsymbol{V}^{(L)}]=&\underset{\boldsymbol{U}^{(1)},\cdots,\boldsymbol{U}^{(L)},\boldsymbol{V}^{(1)},\cdots,\boldsymbol{V}^{(L)}}{\operatorname{argmin}}\frac{1}{2}\sum_{s=1}^{L}\sum_{i=1}^{M}\sum_{j=1}^{N}\boldsymbol{I}_{ij}^{(s)}(\boldsymbol{R}_{ij}^{(s)}-\boldsymbol{U}_i^{(s)\mathrm{T}}\boldsymbol{V}_j^{(s)})^2+\frac{\lambda_u}{2}\sum_{s=1}^{L}\|\boldsymbol{U}^{(s)}\|_F^2\\&+\frac{\lambda_v}{2}\sum_{s=1}^{L}\|\boldsymbol{V}^{(s)}\|_F^2+\alpha\sum_{i=1}^{M}\sum_{s_1=1}^{L}\sum_{s_2\neq s_1}^{L}f_{\mathrm{sim}}(\boldsymbol{U}_i^{(s_1)},\boldsymbol{U}_i^{(s_2)})\\&+\alpha\sum_{i=1}^{N}\sum_{s_1=1}^{L}\sum_{s_2\neq s_1}^{L}f_{\mathrm{sim}}(\boldsymbol{V}_i^{(s_1)},\boldsymbol{V}_i^{(s_2)})\end{aligned}\tag{9}$$

4　模型推断

在模型推断部分，我们通过随机梯度下降算法推断隐因子矩阵 $\boldsymbol{U}$ 和 $\boldsymbol{V}$。对于 SCMF(Lin)，在更新过程中，我们依次遍历每个打分矩阵中的每一个打分对，并且更新与改打分对相关的用户和物品的隐因子向量。给定一个用户物品对 (i,j) 在 s 数据源中的打分 $\boldsymbol{R}_{ij}^{(s)}$，我们计算目标函数关于它的偏导，并沿着偏导方向更新 $\boldsymbol{U}_i^{(s)}$ 和 $\boldsymbol{V}_j^{(s)}$。

$$\boldsymbol{U}_i^{(s)}\leftarrow\boldsymbol{U}_i^{(s)}-\gamma\frac{\partial\boldsymbol{L}}{\partial\boldsymbol{U}_i^{(s)}}=\boldsymbol{U}_i^{(s)}+\gamma\left(\boldsymbol{R}_{ij}^{(s)}-\boldsymbol{U}_i^{(s)\mathrm{T}}\boldsymbol{V}_j^{(s)}-\lambda_u\boldsymbol{U}_i^{(s)}+\alpha\sum_{l\neq s}^{s}\boldsymbol{U}_i^{l}\right)\tag{10}$$

$$\boldsymbol{V}_j^{(s)}\leftarrow\boldsymbol{V}_j^{(s)}-\gamma\frac{\partial\boldsymbol{L}}{\partial\boldsymbol{V}_j^{(s)}}=\boldsymbol{V}_j^{(s)}+\gamma\left(\boldsymbol{R}_{ij}^{(s)}-\boldsymbol{U}_i^{(s)\mathrm{T}}\boldsymbol{V}_j^{(s)}-\lambda_u\boldsymbol{V}_j^{(s)}+\alpha\sum_{l\neq s}^{s}\boldsymbol{V}_j^{l}\right)\tag{11}$$

其中，γ 为学习的步长。

对于 SCMF(MLP)，在使用随机梯度更新参数时我们要还要考虑到 MLP 函数的参数。我们可以通过后向传播(back propagation)[21]的方式来更新 $\phi\leftarrow\phi-\gamma\frac{\partial\boldsymbol{L}}{\partial\phi}$SCMF 学习算法如表 1 所示。

表 1　SCMF 学习算法

SCMF 学习算法：
$[\boldsymbol{U}^{(1)},\cdots,\boldsymbol{U}^{(L)},\boldsymbol{V}^{(1)},\cdots,\boldsymbol{V}^{(L)}]=\mathrm{SCMF}(\boldsymbol{R}^{(1)},\cdots,\boldsymbol{R}^{(L)},\lambda_u,\lambda_v,\alpha,\gamma,T)$
输入：L 个打分矩阵 $\boldsymbol{R}^{(1)},\cdots,\boldsymbol{R}^{(L)}$，正则约束系数 λ_u,λ_v
　　相似度控制参数 α
　　迭代次数 T，步长 γ
输出：用户和物品在每个数据源中的隐因子矩阵
　　$\boldsymbol{U}^{(1)},\cdots,\boldsymbol{U}^{(L)},\boldsymbol{V}^{(1)},\cdots,\boldsymbol{V}^{(L)}$

```
for 1=1 to T do
  for s= 1 to L do                    (遍历所有的源打分矩阵)
    for each pair (i,j) in R^(s) do
      Update U_i^(s) 和 V_j^(s) (φ, if SCMF(MLP))
    end for
  end for
end for
```

5 实验

5.1 数据集

我们在两个数据集上验证我们的模型。第一个数据集使用 MovieLens 和 Netflix 构建的电影评分数据集。我们整合了 Netflix 和 Movielens 两个数据集,根据电影的标题和年份信息我们能够确定两个数据集中的同一部电影,我们能够构建一个电影评分的多源打分矩阵。由于两个数据集合中用户数目的差异过大,我们从 Netflix 中采样了七万多个用户使得两个数据集均衡。数据集的信息描述在表 2 中,该数据集命名为 MovieHetero,这个数据集是在物品端的多场景数据集。

表 2 电影评分数据信息

统计信息	MovieLens	Netflix
电影数	5 871	5 871
用户数	69 258	70 132
打分数	7 891 832	11 658 783

同时我们考虑了一个用户端的多场景数据集,我们从在线社交网络豆瓣网上采集数据[22]。豆瓣网是中国的一个大型的在线社交兴趣网络,用户会在上面发布对电影、书籍、音乐的评分数据。我们在采集的数据中抽取了 10 000 个用户,构建了一个电影—书籍的用户多场景网络,具体的统计信息见表 3。

表 3 Douban 用户多场景数据信息

统计信息	电影	书籍
用户数	10 000	10 000
物品数	25 342	51 204
打分数	2 287 712	832 103

5.2 实验设计描述

我们使用 C 语言实验我们的算法。我们的实验在一台多核机器上的一个单核上运行,机器的 CPU 为 Intel(R) Xeon(R) E5620,2.40GHz,内存为 16GB。我们实验在运行过程中会用到大约 2GB 的内存。

我们采用 RMSE(root mean square error)作为我们的评价指标,其定义如下:

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{(u,v)\in R_{\mathrm{test}}}(R_{uv}-\hat{R}_{uv})^2}{|R_{\mathrm{test}}|}} \tag{12}$$

其中,R_{test}为测试集合,$\hat{R}_{\mathrm{uv}}$为预测分数,R_{uv}为观察到的打分。我们采用 kNN、MF、CMF 作为基准算法,与 SCMF(Lin)和 SCMF(MLP)两个模型进行比较。

在实验训练阶段,模型在整个数据源上学习参数,在评价阶段在各自的源数据上的测试集合上输出预测效果。我们选择隐因子向量的维度为 30。我们在实验中切分 70%的数据作为训练数据(training data),20%的数据作为验证数据(validation data),10%的数据作为测试数据(test data)。我们使用验证数据来确定正则约束系数、学习步长、迭代次数和混合系数,最终通过 10 次重复实验给出平均值。kNN 算法中,我们选择 $k=50$。

5.3 实验结果及分析

5.3.1 整体性能

首先我们分析我们的算法相比其他算法在整体性能上的结果。表 4 和表 5 分别给出了我们在两个数据集上的结果。在两个不同的场景中,我们的算法相对于其他算法都一致的好。在 MovieHetero 的实验中,kNN 算法的性能表现最差,这是因为 kNN 算法基于启发性的规则,因此其性能相比其他基于模型的算法要差。MF 模型在所有基于模型的算法中表现最差,这是因为 MF 算法是工作在单场景模式下的,并没有整合利用其他场景中的信息。CMF 算法相对 MF 算法能够得到一定的改进,这说明整合利用多个场景里的信息的确能够带来性能上的提升。CMF 算法比我们提出的两个模型的效果都要差,这是因为 CMF 基于硬约束的方式,完全没有考虑到不同场景的差异性。我们提出的两个算法中,SCMF(MLP)要优于 SCMF(Lin),这也进一步说明了非线性约束相对于线性约束的必要性。

我们对结果做了显著性检验,使用双边的 t-test。结果说明了我们算法 SCMF(MLP)在显著性水平为 0.01 的情况下,要显著地优于 CMF 算法。

在 Douban 数据集上的结果和 MovieHetero 数据集上的结果表现一致。这说明了我们提出的算法在用户多场景推荐和物品多场景推荐这两类典型的多场景推荐情形下都具有适用性。

表 4　MovieHetero 实验结果

模型	MovieLens	Netflix
kNN	0.810 5	0.841 4
MF	0. 805 1	0.833 4
CMF	0.798 6	0.833 0
SCMF(Lin)	0.793 7	0.831 5
SCMF(MLP)	**0.790 2**	**0.829 5**

表 5　Douban 实验结果

模型	Movie	Book
kNN	0.720 3	0.768 8
MF	0.715 2	0.759 3
CMF	0.710 1	0.755 1
SCMF(Lin)	0.708 3	0.751 5
SCMF(MLP)	**0.701 5**	**0.745 8**

5.3.2　不同稀疏性状况下的表现

由于数据分布的不均衡问题，只有很少打分的用户和物品受到数据稀疏性的影响最大。在这一节我们分析 SCMF 模型在不同的稀疏程度上的性能。我们在电影多场景数据集中分析，具体地给定一个场景，我们将该场景中的电影按照稀疏程度分为三类：打分数在 100 以下的定义为“冷门”类(cold)，打分数在 100 到 1 000 之间的定义为“普通”类(normal)，打分数在 1 000 以上的定义为“热门”类(warm)。在这种定义下，在一个场景中热门的电影有可能在另一个场景中是普通的。整体上看，所有的电影落在了七个区域里。有两个区域是空的，没有电影在一个场景里面热门而在另一个场景中冷门。图 4 展示了通过利用 Netflix 数据集(辅助领域)的信息在 MovieLens 数据集(目标领域)上的 RMSE 的预测结果，图 5 展示了通过利用 MovieLens 数据集(辅助领域)的信息在 Netflix 数据集(目标领域)上的 RMSE 的预测结果。

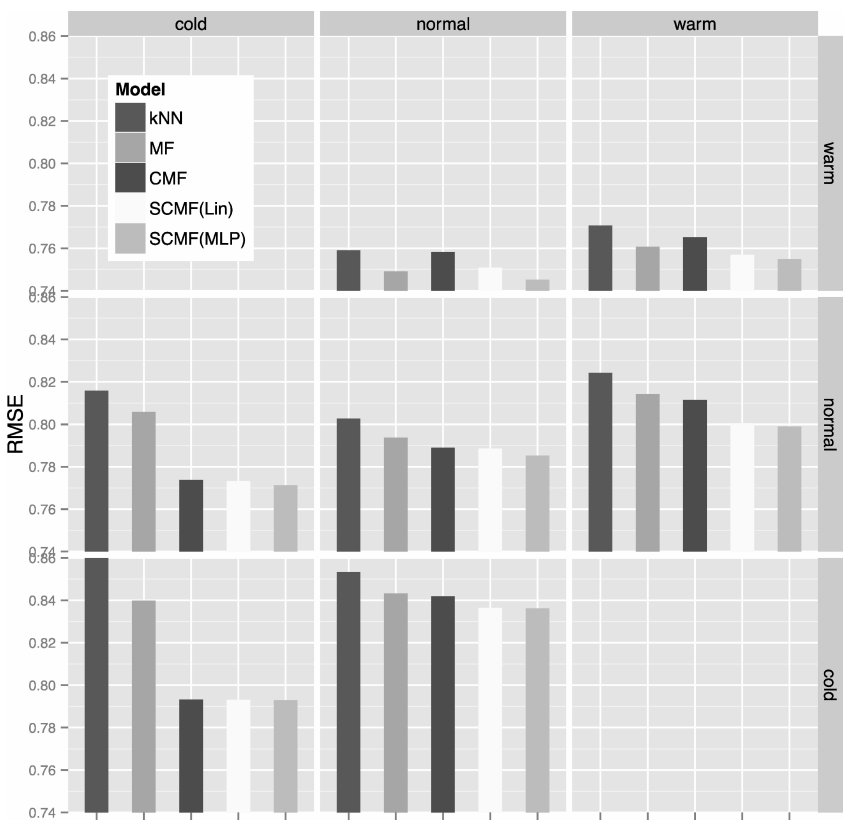

图 4　利用 Netflix 数据集的信息在 MovieLens 数据集上的 RMSE 的预测结果

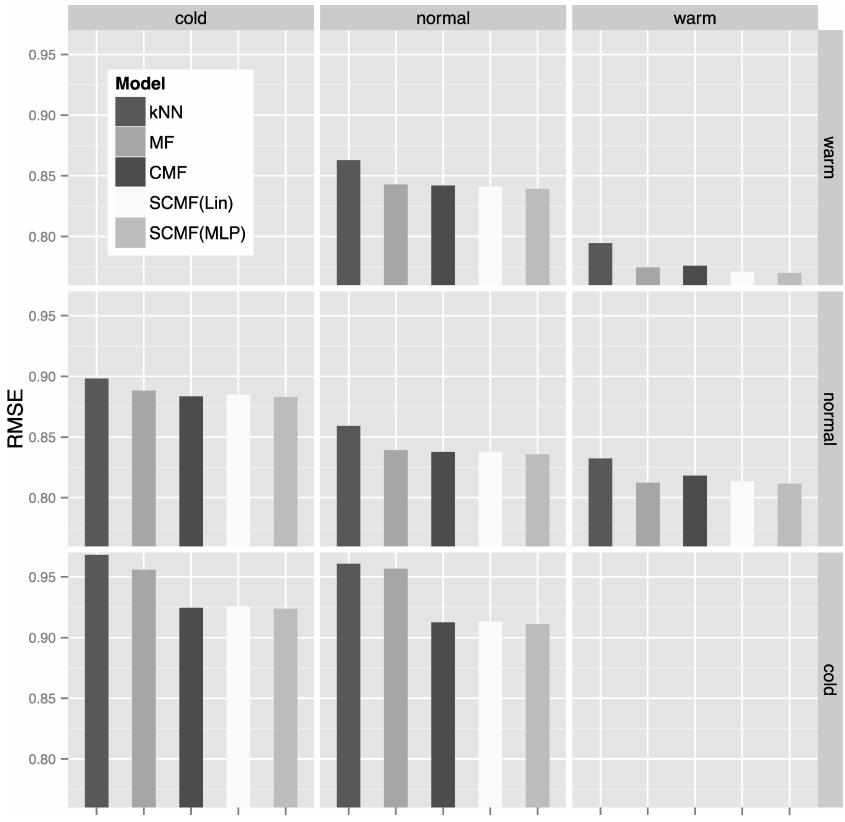

图 5 利用 MovieLens 数据集的信息在 Netflix 数据集上的 RMSE 的预测结果

在所有的情况下,kNN 算法表现最差。在大部分情况下,标准的矩阵分解算法比所有的算法都表现得要差。相比于 MF 算法,CMF 算法在目标领域信息比较稀疏和普通的情况下表现得要好,而当目标领域中信息比较充分的时候性能会下降。这是因为当目标领域的信息很充分的时候,CMF 算法在整合外部信息的时候会损害目标领域的预测性能。而我们的 SCMF 模型在所有稀疏程度的情况下都能够取得比基准算法更好的效果。从实验结果中可以看到,我们的模型同时刻画了处于多个数据源中的相同对象的相似性(相对于 CMF 模型),又较好地抓住了相同的对象处于多个数据源里的差异性(相对于 MF 模型)。

6 总结和未来工作

本文中提出了一个融合多源数据的推荐算法,在矩阵分解模型的基础上,目标函数中加入了约束条件,使得出现在多个源数据中的用户和物品的隐因子向量具有一定的相似性。通过在数据集上的实验发现,我们的算法在推荐精度上要优于传统的算法,尤其是在稀疏性物品上的推荐更是有非常大的改进。

然而,我们的工作中还存在一些需要改进的地方。一个是控制相似程度的参数的选取,通过调参选择结果最优的方式较为费时,可以考虑将模型概率化,从贝叶斯模型的角度使得能够不用调节参数而在学习的过程中自适应地找到最好的参数。另外,我们的模型中通过目标函数的约束项来保证用户和物品在不同数据源中的同质性,而直接分别建模用户和物品的同质性因子和异质性因子似乎更加地直接。在未来的工作中,我们会尝试解决这些问题。

参考文献

[1] Adomavicius Gediminas, Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions [J]. Knowledge and Data Engineering, IEEE Transactions, 2005, 17(6): 734-749.

[2] Sarwar Badrul, et al. Item-based collaborative filtering recommendation algorithms[C]//Proceedings of the 10th International Conference on World Wide Web. ACM, 2001.

[3] Desrosiers, Christian, George Karypis. A comprehensive survey of neighborhood-based recommendation methods [M]. Recommender systems handbook. Springer US, 2011: 107-144.

[4] Koren Yehuda. Factorization meets the neighborhood: a multifaceted collaborative filtering model[C]//Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2008.

[5] Koren Yehuda, Robert Bell, Chris Volinsky. Matrix

factorization techniques for recommender systems[J]. Computer,2008, 42(8): 30-37.
[6] Salakhutdinov Ruslan, Andriy Mnih. Probabilistic matrix factorization[J]. Advances in Neural Information Processing Systems, 2008(20): 1257-1264.
[7] Herlocker Jonathan L, et al. Evaluating collaborative filtering recommender systems[J]. ACM Transactions on Information Systems (TOIS) 2004,22(1): 5-53.
[8] Park Yoon-Joo, Alexander Tuzhilin. The long tail of recommender systems and how to leverage it[C]//Proceedings of the 2008 ACM Conference on Recommender Systems. ACM, 2008.
[9] Ma Hao, Irwin King, Michael R, Liu. Learning to recommend with social trust ensemble[C]//Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information RetrievAL. ACM, 2009.
[10] Koenigstein Noam, Gideon Dror, Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy [C]//Proceedings of the 5th ACM Conference on Recommender Systems. ACM, 2011.
[11] Chen, Tianqi, et al. SVDFeature: a toolkit for feature-based collaborative filtering[J]. The Journal of Machine Learning Research, 2012, 13(1): 3619-3622.
[12] Yu, Xiao, et al. Personalized entity recommendation: a heterogeneous information network approach[C]// Proceedings of the 7th ACM International Conference on Web Search and Data Mining. ACM, 2014.
[13] Jamali, Mohsen, Laks Lakshmanan. HeteroMF: recommendation in heterogeneous information networks using context dependent factor models[C]// Proceedings of the 22nd International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2013.
[14] Li C Y, Lin S D. Matching users and items across domains to improve the recommendation quality[C]// Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2014: 801-810.
[15] Ozsoy, Makbule Gulcin, Faruk Polat, Reda Alhajj. Modeling individuals and making recommendations using multiple social networks[C]//Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015. ACM, 2015.
[16] Berk Shlomo, Tsvi Kuflik, Francesco Ricci. Cross-domain mediation in collaborative filtering[M]. User Modeling 2007. Springer Berlin Heidelberg, 2007: 355-359.
[17] Pan Weike, et al. Transfer learning in collaborative filtering for sparsity reduction[C]//Proceedings of the 24rd AAAI Conference on Artificial Intelligence, 2010.
[18] Singh Ajit P, Geoffrey J Gordon. Relational learning via collective matrix factorization[C]//Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2008.
[19] Tong Man, Huawei Shen, Junming Huang, Xueqi Cheng. Context-adaptive matrix factorization for multi-context recommendation [C]//Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015), Melbourne, Australia. October 2015: 901-910.
[20] Tong Man, Huawei Shen, Xiaolong Jin, Xueqi Cheng. Cross-domain recommendation: an embedding and mapping approach[C]//Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017), Melbourne, Australia. August 2017: 2464-2470.
[21] Werbos, Paul J. Backpropagation through time: what it does and how to do it[C]//Proceedings of the IEEE, 1990, 78(10): 1550-1560.
[22] Huang, Junming, et al. Exploring social influence via posterior effect of word-of-mouth recommendations[C]// Proceedings of the 5th ACM International Conference on Web Search and Data Mining. ACM, 2012.
[23] Hu, Liang, et al. Personalized recommendation via cross-domain triadic factorization[C]//Proceedings of the 22nd International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2013.
[24] Rendle, Steffen, et al. BPR: bayesian personalized ranking from implicitfeedback [C]//Proceedings of the 25th Conference on Uncertainty in Artificial intelligence. AUAI Press, 2009.
[25] Rumelhart David E, Geoffrey E Hinton, Ronald J Williams. Learning representations by back-propagating errors[J]. Cognitive modeling, 1988,5(3): 533-536.

满彤(1989—),博士,主要研究领域为推荐系统,数据挖掘。
E-mail: supermt@gmail.com

黄俊铭(1984—),博士,主要研究领域为信息传播,社交网络分析。
E-mail: mail@junminghuang.com

沈华伟(1982—),通信作者,博士,副研究员,主要研究领域为网络科学、社会网络分析、数据挖掘。
E-mail: shenhuawei@ict.ac.cn