计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(8):135−143 [doi: 10.15888/j.cnki.csa.007491]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于聚类和奖惩用户模型的协同过滤算法①

吴青洋, 程　旭, 邓程鹏, 丁浩轩, 张　宏, 林胜海

(中汽数据有限公司, 天津 300393)
通讯作者: 吴青洋, E-mail: 287366123@qq.com

摘　要: 根据用户体验为其推荐感兴趣的项目是推荐系统中最重要的问题. 本文提出了一种新的易于实现的 CBCF (Clustering-Based CF) 算法, 该算法基于激励/惩罚用户 (IPU) 模型进行推荐. 本文旨在通过 IPU 模型深入研究用户间偏好的差异来提高准确率、召回率和 F1-score 方面的性能. 本文提出了一个约束优化问题, 目标是在给定的精度下最大限度地提高召回率 (或 F1-score). 为此, 根据实际评分数据和皮尔逊相关系数, 将用户分为若干用户簇, 然后根据同一用户簇的偏好倾向, 对每个项目进行奖励/处罚. 实验结果表明, 本文提出的算法在给定准确率的条件下, 召回率可以显著提高 50% 左右.

关键词: 聚类; 协同过滤推荐; F1-score; 激励/惩罚用户模型; 皮尔逊相关系数; 推荐系统

引用格式: 吴青洋,程旭,邓程鹏,丁浩轩,张宏,林胜海.基于聚类和奖惩用户模型的协同过滤算法.计算机系统应用,2020,29(8):135–143. http://www.c-s-a.org.cn/1003-3254/7491.html

Collaborative Filtering Algorithm Based on Clustering and Incentive/Penalty User Model

WU Qing-Yang, CHENG Xu, DENG Cheng-Peng, DING Hao-Xuan, ZHANG Hong, LIN Sheng-Hai

(Automotive Data of China (Tianjin) Co. Ltd, Tianjin 300393, China)

Abstract: Giving or recommending appropriate content based on the quality of experience is the most important in recommender systems. This study proposes a new CBCF (Clustering-Based CF) method using an Incentivized/Penalized User (IPU) model, which is thus easy to implement. The purpose of this study is to improve recommendation performance of accuracy, recall and F1-score by studying the differences of users' preferences through IPU model. This study formulates a constrained optimization problem in which we aim to maximize the recall (or equivalently F1-score) for a given precision. To this end, users are divided into several clusters based on the actual rating data and Pearson correlation coefficient. Afterward, we give each item an incentive/penalty according to the preference tendency by users within the same cluster. Experiments show that under the condition of given accuracy, the recall rate of the proposed algorithm can be improved by about 50%.

Key words: clustering; collaborative filtering; F1-score; incentivized/penalized user model; Pearson correlation coefficient; recommender system

由于互联网不断产生大量视频、音频、文章等, 人们很难有效地找到自己喜欢的事物, 个性化推荐系统能够帮助人们快速从大量信息中作出选择、提供建议、辅助决策[1]. 阿里, 京东等公司通过使用推荐系统吸引了大量的用户, 并通过推荐系统提供的个性化服务, 创造了惊人的销售业绩.

个性化推荐、基于内容的推荐和基于知识的推荐已经得到了广泛应用, 其中协同过滤 (CF) 是推荐系统

① 收稿时间: 2019-11-12; 修改时间: 2019-12-23, 2020-01-07; 采用时间: 2020-01-14; csa 在线出版时间: 2020-07-29

中最突出、最流行的技术之一[2]. 协同过滤算法一般分为基于邻域的协同过滤和基于模型的协同过滤. 基于模型的协同过滤, 通过使用大量数据来训练模型, 然后使用该模型预测用户的偏好. 加权 λ 正则化的交替最小二乘法 (ALS-WR) 是基于模型的 CF 的一个经典案例, ALS-WR 是基于矩阵因子分解算法实现的, 并且能够很好地解决数据的稀疏性和可扩展性问题[3].

基于模型的 CF 在提高预测精度以及应对数据稀疏性方面优势明显. 但它有一些缺点, 如构建模型的成本很高[2]. 基于邻域的 CF 不需要构建特定的模型, 而是使用用户评分矩阵来计算用户或项目之间的相似性. 因此, 基于邻域的 CF 实现起来更容易. 但是, 它也有一些缺点, 如十分依赖用户的评分、当数据很稀疏时预测精度急剧下降以及无法为新用户进行推荐[2]. 基于邻域的 CF 算法又分为基于用户的 CF 和基于项目的 CF. 基于用户的 CF 和基于项目的 CF 算法的本质是根据评分计算用户相似性和项目相似性, 在找到相似的用户 (称为邻居) 后, 基于用户的 CF 将邻居们最喜爱而自己不熟悉的前 N 个项目进行推荐. 当用户数量远远大于项目数量时, 基于用户的 CF 可扩展性较差. 不少学者曾尝试使用基于项目的 CF 来解决可扩展性问题, 但是当用户和项目的数量很大时, 仍然不能完全解决这个问题. 尽管 CF 有这些不足, 但它仍然是最具代表性的推荐算法.

文献 [4] 在致力于降低平均绝对误差 (MAE) 或均方根误差 (RMSE) 方面, 对 CF 算法进行了大量的研究. 然而, 对推荐系统来说仅靠降低 MAE 或 RMSE 数值, 并不能从本质上提高推荐的准确性. 假设两个推荐系统具有相同的评分预测 MAE 或 RMSE. 值得注意的是它们在用户体验 (UX) 方面可能不同, 因为一个推荐系统可能推荐一个项目, 而另一个推荐系统没有推荐该项目. 针对上述不足, 与用户体验相关的性能指标, 如查准率、召回率和 F1-score 得到了广泛应用.

潘多拉互联网电台、Netflix 和 Artsy 基于聚类的推荐算法分别开发了音乐基因组项目、微电影和艺术基因组项目. 这些基于聚类的推荐算法取得了令人满意的效果, 但是聚类的处理成本很高. 例如, 就音乐基因组项目而言, 音乐家分析每首歌曲的过程通常需要 20 到 30 分钟.

综上所述, 基于邻域的协同过滤算法存在数据稀疏性以及冷启动问题, 基于模型的协同过滤算法在提高预测精度以及应对数据稀疏性方面优势明显, 但构建模型的成本很高. Netflix、Artsy 基于聚类的推荐算法取得了令人满意的效果, 因此本文采用聚类的推荐算法, 针对聚类的处理成本较高, 本文设计了一种处理成本低、只需要用户给出评分, 简单易于实现的聚类的算法; 为了提高推荐准确率, 根据实际评分数据和皮尔逊相关系数, 将用户分为若干用户簇, 并深入研究了用户与用户簇之间的偏好的差异, 根据同一用户簇中用户的偏好倾向, 对每个项目进行激励/惩罚, 即本文通过使用激励/惩罚用户模型 (IPU) 的 CBCF 算法, 在准确率、召回率和 F1-score 方面来提高推荐系统的性能.

1 相关研究

本文提出的算法涉及推荐系统中的 CF 算法、聚类算法、基于聚类的推荐系统等研究领域, 对推荐系统的性能指标诸如准确率和召回率等进行了研究分析, 并总结了基于 CF 算法的偏好预测以及两种聚类算法.

1.1 CF 算法

CF 是推荐系统最常用的技术之一, 但在数据稀疏和冷启动等方面存在不足[5]. 如果用户评分矩阵过于稀疏, 那么预测评分就会不准确. 此外, CF 很难对新用户或项目进行预测评分. 解决这两个问题有很大的挑战[6]. 文献 [4] 在如何提高 CF 推荐系统的预测精度上进行了研究.

1.2 聚类算法

聚类已广泛应用于各种数据挖掘应用: 如 K-means 以及文献 [7] 提出的监控游戏粘性 DBSCAN 聚类算法. 文献 [8] 提出了一种新的基于熵的目标函数来聚类不同类型的图片. 为了满足并行处理系统的实时性要求, 文献 [9] 中提出了一种改进的一维数据均值聚类算法.

1.3 基于聚类的推荐系统

文献 [10–12] 在通过聚类算法提高推荐的准确性方面进行了深入研究. 文献 [10] 中, CF 和基于内容的推荐分别用于查找相似的用户和项目, 并进行聚类, 然后对目标用户进行个性化推荐. 结果表明在准确率、召回率和 F1 score 方面的表现有所改善. 文献 [11] 提出对每组数据进行矩阵分解之前先进行聚类处理. 文献 [12] 对使用 K-means、自组织映射 (SOM) 和模糊 C 均值 (FCM) 聚类算法应用于基于用户 CF 的性能进行研究. 结果表明, 与 K 均值和 SOM 聚类算法相比, 使用 FCM 聚类算法的基于用户的 CF 具有更好的性能.

1.4 性能指标分析及偏好预测方法

文献 [13] 研究了广泛应用于评价推荐系统优劣的

性能指标, 如准确率、召回率和 F1-score 等. 使用 CF 进行偏好预测的算法分为基于邻域的推荐算法和基于模型的推荐算法. 基于邻域的推荐算法直接利用大量历史数据来预测目标项目的评分, 并为活跃用户进行推荐. 基于邻域的推荐算法进行推荐过程中需要将所有数据加载到内存中, 并在数据上实现特定的算法. 基于模型的推荐算法通过基于已知数据利用数据挖掘的方法来建立预测模型, 建立好预测模型后, 在推荐过程中就不再需要历史数据了[14].

本文研究了基于邻域的 CBCF 算法, 尽管基于模型的方法在预测速度和可扩展性方面具有优势, 但在灵活性和预测质量等方面存在不足, 建立模型通常是一个耗费时间和资源的过程, 建立模型的方式对预测质量的影响较大.

1.5 聚类

在 SOM、K-means、FCM 和谱聚类等聚类算法中, 本文选择了谱聚类和 FCM 聚类算法, 因为这两种算法能取得令人满意的效果. 下面简单阐述一下这两种算法.

谱聚类使用了关联矩阵的特征向量来进行聚类. 两个对象之间的相似度越高, 这两个对象之间的关联值越大. 高斯相似度函数用于计算两个对象之间的相似度, 常用于构造关联矩阵, 高斯相似函数 $s(x_i,x_j)=e^{\frac{-\|x_i-x_j\|^2}{2\sigma^2}}$. 其中, σ 控制邻居范围的大小[15]. 得到关联矩阵后, 对关联矩阵的特征向量来进行聚类, 最后谱聚类根据特征向量来进行聚类. 谱聚类实现简单, 可以通过标准的线性代数软件进行求解, 而且效果明显也优于传统的聚类算法 (如 K-means 算法)[15].

FCM 聚类[16] 通过系数 w_{ij}^m 将对象 x_i 划分到簇 c_j, 使每个对象成为具有不同模糊隶属度的所有聚类的成员, 其中 m 是控制集群模糊程度的超参数. m 越大, 簇越模糊. FCM 聚类首先在给定多个聚类的情况下随机初始化每个聚类的中心点. 然后, 重复以下两个步骤直到两次迭代之间系数的变化小于给定的阈值. (1) 计算每个簇的质心; (2) 计算每个点在簇中的系数.

2 基于聚类和奖惩用户模型的推荐算法

2.1 问题定义

本文的创新在于当 MAE 或 RMSE 相同的情况下, 可以做出是否推荐某个项目的正确决策, 来提升用户体验. 例如, 假设用户 A 对项目 B 的实际评分为 4.2, 两个推荐系统分别预测用户 A 对项目 B 的评分偏好为 3.8 和 4.6. 预测评分大于 4.0 的项目将推荐给用户, 这两个推荐系统的 MAE 是相同的, 但只有后一个系统会推荐该项目. 为了提升用户体验, 本文根据用户的偏好倾向对每一个项目进行激励或惩罚. 为此, 将用户分为若干簇, 并根据用户所属的簇的情况来决定对项目的奖惩.

图 1 显示了使用 IPU 模型的 CBCF 算法的示例, 假设有 2 个项目和 4 个用户簇, 假设用户被分成 4 个簇, 即 C_1、C_2、C_3 和 C_4. 从图 1 可以看出, 用户 $u1$、$u2$、$u6$ 和 $u17$ 属于簇 C_1. 其中, 实心方形项目和实心圆形项目分别表示测试数据和训练数据. $\hat{r}_{u,i}$ 和 $r_{u,i}$ 分别为用户 u 对项目 i 的预测评分和实际评分, 其中基于邻域的 CF 和基于模型的 CF 可用于评分预测. 如图 1 所示, 用户 17 对项目 1 实际的评分 $r_{u17,i1}=4.0$ 及其预测的评分 $\hat{r}_{u17,i1}=3.9$. 用户 u 已经评分的项目用红色实心表示, 没有评分的项目用空心表示. 例如, 在簇 C_1 中, 用户 $u1$、$u2$ 和 $u17$ 对 $i1$ 的评分分别为 5.0、5.0 和 4.0, 即 $r_{u1,i1}=5.0$、$r_{u2,i1}=5.0$ 和 $r_{u17,i1}=4.0$. 用户 $u1$、$u2$ 和 $u6$ 对 $i2$ 的评分分别为 5.0、4.0 和 3.0, 即 $r_{u1,i2}=5.0$、$r_{u2,i2}=4.0$ 和 $r_{u6,i2}=3.0$. $\overline{C}_c^i$ 表示为用户簇 Cc 中用户对项目 i 的平均偏好. $\overline{C}_c^i$ 可以表示为:

$$\overline{C}_c^i=\frac{\sum_{u\in U_{i,c}} r_{u,i}}{|U_{i,c}|} \tag{1}$$

其中, $U_{i,c}$ 是用户簇 C_c 中对项目 i 进行评分的一组用户. 如图 1 所示, 用户簇 $C1$ 对 $i1$ 的平均偏好 $\overline{C}_1^{i1}$ 为 4.67.

根据每个用户簇 $\overline{C}_c^i$, 使用 IPU 模型来决定某个项目是否推荐给用户 u. 具体推荐策略为, 当 $\overline{C}_c^i$ 的值足够大时, 即 $\overline{C}_c^i\geq\gamma$, 给予项目 i 激励, 其中 γ>0 表示待优化的参数. 当 $\overline{C}_c^i<\gamma$, 给予项目 i 惩罚, 系统参数 α 和 β 分别用作惩罚和激励的阈值, 并设置为正数, 其中 $\alpha\geq\beta$. 例如, 假设 α=4.5, β=3.5, γ=3.0. 在图 1 中, $i1$ 将被推荐给 $u19$, 但如果 $i1$ 和 $i2$(即 $\hat{r}_{u19,i1}$ 和 $\hat{r}_{u19,i2}$) 的预测偏好分别为 3.8 和 4.2, 则 $i2$ 将不被推荐给 $u19$. 因为 $\overline{C}_3^{i1}(=4.33)$ 大于 $\gamma(=3.0)$, $\hat{r}_{u19,i1}(=3.8)$ 也大于 $\beta(=3.5)$ 然而, 在 $i2$ 的情况下, $u19$ 不被推荐, 因为 $\overline{C}_3^{i2}(=2.33)$ 小于 γ, $\hat{r}_{u19,i2}<\alpha$. 简而言之, 根据用户簇获得的偏好倾向来改变推荐决策.

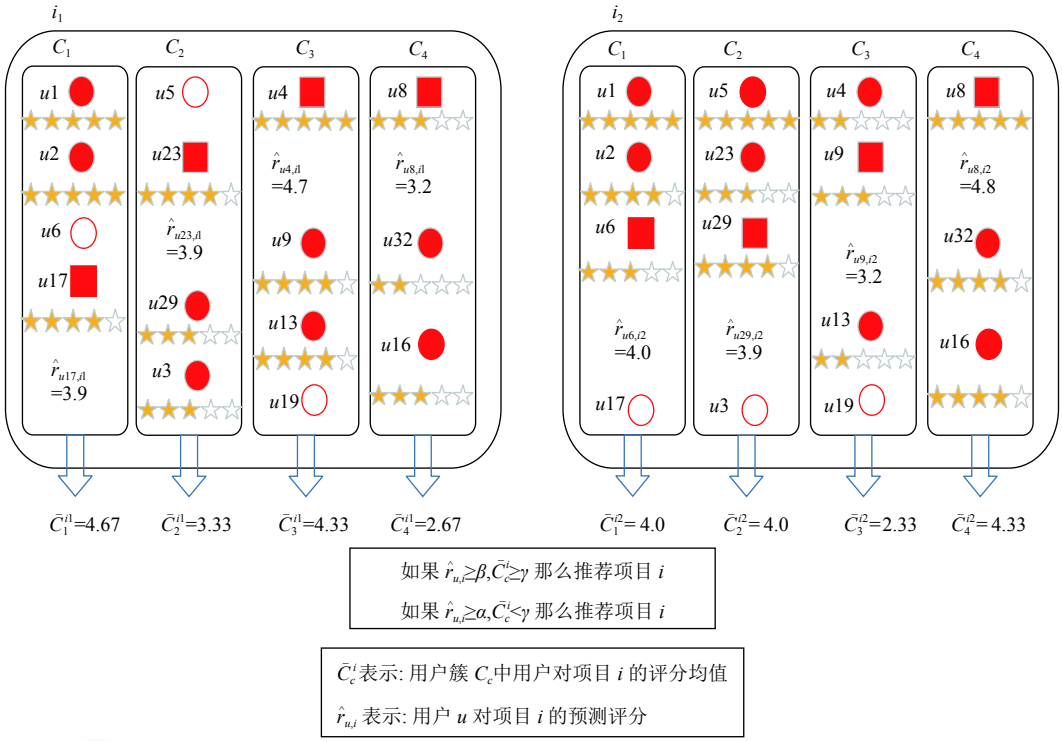

图 1 基于 IPU 模型的 CBCF 算法的例子, 假设有 2 个项目和 4 个用户簇, 实心方形项目和实心圆形项目分别表示测试数据和训练数据

算法 1. 使用 IPU 模型的 CBCF 算法

```
1.  if C̄_c^i≥γ then
2.    if r̂_{u,i}≥β then
3.      将项目 i 推荐给用户 u
4.    else 项目 i 不被推荐
5.  else
6.    if r̂_{u,i}≥α then
7.      将项目 i 推荐给用户 u
8.      else 项目 i 不被推荐
9.  end
```

算法 1 描述了使用 IPU 模型 CBCF 算法, 从算法 1 中可以看出, 当$\bar{C}_c^i \geq \gamma$, 时, 只有当预测评分大于 β 才进行推荐. 如果$\bar{C}_c^i < \gamma$,, 则仅当预测评分大于 α 才进行推荐.

如前所述, 准确率、召回率和 F1-score 作为性能评估的指标, 这 3 个性能指标可以表示为真正类 (TP)、真负类 (TN)、假正类 (FP) 和假负类 (FN) 的函数. 假设预测条件为真. 如果条件实际为真 (或假), 则为 TP (或 FP), 假设预测条件为假, 如果条件实际上为真 (或假), 则为 FN (或 TN). 对于给定的用户 u 和项目 i, TP、TN、FP 和 FN 依赖于 α、β 和 γ, 因此:

$$\begin{cases} f_{\mathrm{TP}}^{u,i}(\alpha,\beta,\gamma,\delta_{\mathrm{pref}}) = I_{[\gamma,\infty)}(\bar{C}_c^{u,i})\cdot I_{[\beta,\infty)}(\hat{r}_{u,i})\cdot I_{[\delta_{\mathrm{pref}},\infty)}(r_{u,i}) + I_{(0,\gamma)}(\bar{C}_c^{u,i})\cdot I_{[\alpha,\infty)}(\hat{r}_{u,i})\cdot I_{[\delta_{\mathrm{pref}},\infty)}(r_{u,i}) \\ f_{\mathrm{FP}}^{u,i}(\alpha,\beta,\gamma,\delta_{\mathrm{pref}}) = I_{[\gamma,\infty)}(\bar{C}_c^{u,i})\cdot I_{[\beta,\infty)}(\hat{r}_{u,i})\cdot I_{(0,\delta_{\mathrm{pref}})}(r_{u,i}) + I_{(0,\gamma)}\bar{C}_c^{u,i})\cdot I_{[\alpha,\infty)}(\hat{r}_{u,i})\cdot I_{(0,\delta_{\mathrm{pref}})}(r_{u,i}) \\ f_{\mathrm{FN}}^{u,i}(\alpha,\beta,\gamma,\delta_{\mathrm{pref}}) = I_{[\gamma,\infty)}(\bar{C}_c^{u,i})\cdot I_{(0,\beta)}(\hat{r}_{u,i})\cdot I_{[\delta_{\mathrm{pref}},\infty)}(r_{u,i}) + I_{(0,\gamma)}(\bar{C}_c^{u,i})\cdot I_{(0,\alpha)}(\hat{r}_{u,i})\cdot I_{[\delta_{pref},\infty)}(r_{u,i}) \\ f_{\mathrm{TN}}^{u,i}(\alpha,\beta,\gamma,\delta_{\mathrm{pref}}) = I_{[\gamma,\infty)}(\bar{C}_c^{u,i})\cdot I_{(0,\beta)}(\hat{r}_{u,i})\cdot I_{(0,\delta_{pref})}(r_{u,i}) + I_{(0,\gamma)}(\bar{C}_c^{u,i})\cdot I_{(0,\alpha)}(\hat{r}_{u,i})\cdot I_{(0,\delta_{pref})}(r_{u,i}) \end{cases} \tag{2}$$

其中, $IA(x)$ 是集合 A 函数, δ_{pref} 是确定用户是否真正喜欢相应项目的阈值, 其中 δ_{pref} 通常设置为 4.0 (用户评分满分为 5.0) 或 8.0 (用户评分满分为 10.0). 如果 $\bar{C}_c^{u,i} \geq \gamma$, $\hat{r}_{u,i} \geq \beta$, $r_{u,i} \geq \delta_{\mathrm{pref}}$ 则 $f_{\mathrm{TP}}^{u,i} = 1$; 如果$\bar{C}_c^{u,i} < \gamma$, $\hat{r}_{u,i} \geq \alpha$, $r_{u,i} \geq \delta_{\mathrm{pref}}$则 $f_{\mathrm{TP}}^{u,i} = 1$; 反之$f_{\mathrm{TP}}^{u,i} = 0$. 类似的, 如果 $\bar{C}_c^{u,i} \geq \gamma$, $\hat{r}_{u,i} \geq \beta$, $r_{u,i} < \delta_{\mathrm{pref}}$ 则$f_{\mathrm{FP}}^{u,i} = 1$; 如果$\bar{C}_c^{u,i} < \gamma$, $\hat{r}_{u,i} \geq \alpha$, $r_{u,i} < \delta_{\mathrm{pref}}$ 则$f_{\mathrm{FP}}^{u,i} = 1$, 反之$f_{\mathrm{FP}}^{u,i} = 0$ 而且, 如果 $\bar{C}_c^{u,i} \geq \gamma$, $\hat{r}_{u,i} < \beta$, $r_{u,i} \geq \delta_{\mathrm{pref}}$ 则$f_{\mathrm{FN}}^{u,i} = 1$; 如果$\bar{C}_c^{u,i} < \gamma$, $\hat{r}_{u,i} < \alpha$, $r_{u,i} \geq \delta_{\mathrm{pref}}$则$f_{\mathrm{FN}}^{u,i} = 1$; 反之 $f_{\mathrm{FN}}^{u,i} = 0$. $f_{\mathrm{TN}}^{u,i}$的计算方法与上述方法类似. 基于式 (2), 准确率和召回率为:

$$\begin{cases} precision(\alpha,\beta,\gamma,\delta_{\text{pref}}) = \dfrac{\sum_{(u,i)\in T} f_{\text{TP}}^{u,i}(\alpha,\beta,\gamma,\delta_{\text{pref}})}{\sum_{(u,i)\in T} f_{\text{TP}}^{u,i}(\alpha,\beta,\gamma,\delta_{\text{pref}}) + \sum_{(u,i)\in T} f_{\text{FP}}^{u,i}(\alpha,\beta,\gamma,\delta_{\text{pref}})} \\ recall(\alpha,\beta,\gamma,\delta_{\text{pref}}) = \dfrac{\sum_{(u,i)\in T} f_{\text{TP}}^{u,i}(\alpha,\beta,\gamma,\delta_{\text{pref}})}{\sum_{(u,i)\in T} f_{\text{TP}}^{u,i}(\alpha,\beta,\gamma,\delta_{\text{pref}}) + \sum_{(u,i)\in T} f_{\text{FN}}^{u,i}(\alpha,\beta,\gamma,\delta_{\text{pref}})} \end{cases} \tag{3}$$

其中, T 表示用于计算准确率和召回率的测试数据集. F1-score 计算公式如下:

$$F_1(\alpha,\beta,\gamma,\delta_{\text{pref}}) = \frac{2precision \times recall}{precision + recall} \tag{4}$$

回顾图 1 中的例子, 其中 α=4.5, β=3.5, γ=3.0. 实心方形项目表示测试数据. 假设评分超过 4 星的项目是用户所感兴趣的, 即δ_{pref} = 4.0, 是推荐系统中的一个典型假设[17]. 那么, 项目 $i1$ 应该推荐给用户 $u17$, 而不能推荐给用户 $u8$. 用户 $u29$ 和 $u8$ 实际上对项目 $i2$ 很感兴趣. 根据图 1 中的测试数据集, TP、TN、FP 和 FN 总结在表 1 中. 本文将不使用聚类算法的 CF 与本文提出的算法进行比较, 为此, 假设 γ=0, 并修改推荐策略, 只有当预测的偏好$\hat{r}_u^i$不小于 4.0 时, 才推荐项目 i. 表 1 中描述了 TP、TN、FP 和 FN, 从表 1 可知, 当 γ=0 时, $u4$ 对项目 $i1$ 感兴趣, $u6$、$u8$ 对项目 $i2$ 感兴趣, 当 γ=3.0 时, $u17$、$u4$ 对项目 $i1$ 感兴趣, $u6$、$u29$、$u8$ 对项目 $i2$ 感兴趣, 即 $u17$ 实际上对项目 $i1$ 很感兴趣, $u29$ 实际上对项目 $i2$ 很感兴趣. 利用表 1 的结果, 本文计算 γ=0 和 γ=3.0 这两种情况的准确率和召回率, 如表 1 所示.

表 1 当γ = 0和γ = 3时, TP、TN、FP 和 FN 的例子

		项目 $i1$	项目 $i2$
γ = 0 (基于CF不使用聚类算法)	推荐项目	$u_4 \Rightarrow$ TP	$u_6 \Rightarrow$ FP $u_8 \Rightarrow$ TP
	不推荐项目	$u_{17} \Rightarrow$ FN $u_{23} \Rightarrow$ FN $u_8 \Rightarrow$ TN	$u_9 \Rightarrow$ TN $u_{29} \Rightarrow$ FN
γ = 3.0 (本文提出的算法)	推荐项目	$u_{17} \Rightarrow$ TP $u_4 \Rightarrow$ TP	$u_6 \Rightarrow$ FP $u_{29} \Rightarrow$ TP $u_8 \Rightarrow$ TP
	不推荐项目	$u_{23} \Rightarrow$ FN $u_8 \Rightarrow$ TN	$u_9 \Rightarrow$ TN

(1) γ=0(不使用聚类的 CF 算法): 从表 1 可以看出, TP=2, FP=1, FN=3. 因此, 使用式 (3) 计算准确率为 2/3, 召回率为 2/5.

(2) γ=3.0(本文提出的算法): 假设α=4.5 且β=3.5. 从表 1 可以得出, TP=4, FP=1, FN=1, 使用式 (3) 计算准确率为 4/5, 召回率为 4/5.

因此, 当用户分为多个簇, 使用 IPU 模型, 并适当调整系统参数 α、β 和 γ, 可以显著地提高准确率和召回率.

2.2 公式化

值得注意的是, 准确率、召回率和 F1-score 随 α、β 和 γ 的变化而变化. 因此, 本文的目标是找到最优值 α、β 和 γ, 从而最大化 F1-score(或召回率). 因此, 本文提出了一个新的约束优化问题 (由于参数 δ_{pref} 通常设置为某个值, 为了简化符号, 则 δ_{pref} 将从每个函数的参数中删除) 如下所示:

$$\begin{cases} \underset{\alpha,\beta,\gamma}{\text{maximize}} \quad F_1(\alpha,\beta,\gamma) \text{ or } recall(\alpha,\beta,\gamma) \\ \text{当} \quad precision(\alpha,\beta,\gamma) \geq \delta_{precision}, \alpha \geq \beta \end{cases} \tag{5}$$

其中, $\delta_{precision}$ 是一个预先定义的阈值, 并根据不同类型的推荐系统适当地调整该值. 根据不同目的对式 (5) 进行修改也很容易. 例如, 当 $recall(\alpha,\beta,\gamma)\geq\delta$ 情况下, 最大化 $precision(\alpha,\beta,\gamma)$, 或 $precision(\alpha,\beta,\gamma)\geq\delta$ 的情况下, 最大化 $recall(\alpha,\beta,\gamma)$, 找到最优的 α、β 和 γ, 其中 δ_{recall} 是一个预先定义的召回率阈值. 因此, 使用 IPU 模型的 CBCF 算法, 通过用户聚类以及找到参数 α、β 和 γ 的最优值可以提高准确率、召回率和 F1-score.

2.3 基于 IPU 模型的 CBCF 算法

CBCF 算法通过用户聚类以及使用 IPU 模型分析用户间的偏好倾向从而进行推荐. 使用 IPU 模型的 CBCF 算法的核心是根据$\overline{C}_c^i$ (用户簇Cc中对项目 i 的偏好的均值) 的结果对每个项目进行激励或惩罚. 由于评分矩阵R_{CBCF}中存在用户未评分项目, 因此无法准确计算用户向量之间的欧几里得距离 (即R_{CBCF}中的行向量). 因此, 本文使用皮尔逊相关系数 (PCC). PCC 通过计算两个用户的共同评分之间的相关性来衡量其相似性, 两个用户 $u1$ 和 $u2$ 之间的相似度 $s(u1, u2)$ 为:

$$s(u1,u2) = \frac{\sum_{i\in I_{u1}\cap I_{u2}} (r_{u1,i} - \bar{r}_{u1})\cdot(r_{u2,i} - \bar{r}_{u2})}{\sqrt{\sum_{i\in I_{u1}\cap I_{u2}} (r_{u1,i} - \bar{r}_{u1})^2} \cdot \sqrt{\sum_{i\in I_{u1}\cap I_{u2}} (r_{u2,i} - \bar{r}_{u2})^2}} \tag{6}$$

其中, I_{u1}和I_{u2}分别是 $u1$ 和 $u2$ 评分的项目集, $\bar{r}_{u1}$和$\bar{r}_{u2}$分别是 $u1$, $u2$ 用户评分项目交集$I_{u1}\cap I_{u2}$ 上的评分均值. $s(u1, u2)$ 的范围是–1 到 1.

算法 2. 使用 IPU 模型的 CBCF 算法

1. 用户簇$C\in\{C_1,\cdots,Cc\}$;
2. 初始化 n, x, m 的用户评分矩阵R_{CBCF};
3. $\hat{R}\leftarrow R_{\text{CBCF}}$ 预测评分;
4. 初始化阈值 α,β,γ;
5. for $u\leftarrow$1 to n do
6. $I_u\leftarrow$用户测试集中缺少评分的项目;
7. u;
8. $\hat{r}_{u,Iu}\leftarrow$项目I_u的预测评分;
9. for $i\leftarrow$1 to $|I_u|$ do
10. $C_{\text{tmp}}\leftarrow$用户 u 所属的用户簇;
11. $\overline{C}^{i}_{\text{tmp}}\leftarrow$用户簇中的用户对项目 i 的评分均值;
12. if $\hat{r}_{u,i}\geq\alpha$ then
13. 将项目 i 推荐给用户 u;
14. else if $\hat{r}_{u,i}\geq\beta$ && $\overline{C}^{i}_{\text{tmp}}\geq\gamma$ then
15. 将项目 i 推荐给用户 u;
16. else 项目 i 不被推荐;
17. end
18.end

下面着重说明算法 2 的整个流程: 首先, 通过聚类的结果获得聚类集合 C, 并初始化 n, x, m 评分矩阵R_{CBCF} (参见算法 2 中的第 1、2 行). 接下来, 使用基于邻域的算法预进行偏好预测, 并将预测结果保存在$\hat{r}$中 (参阅第 3 行). 更具体地说, 基于用户/项目的 CF 算法用于评估本文提出的 CBCF 算法的性能. 通过求解式 (5) 中的优化问题, 确定阈值 α、β 和γ. 在 for 循环中, I_u是用户 u 的测试集中缺少评分的项, $\hat{r}_{u,I_u}$是I_u中的预测评分, 其中$|I_u|$表示 I_u的基数. 下面通过 α、β 和γ来决策是否推荐某些项目. 当$\hat{r}_{u,i}\geq\alpha$, 时, 将项目 i 推荐给用户 u, 而不需要考虑算法 1 中提到的阈值γ(参考算法 2 中的第 11、12 行). 当$\hat{r}_{u,i}<\alpha$, 时, 那就需要考虑阈值γ以及$\overline{C}^{i}_{\text{tmp}}$, 其中$\overline{C}^{i}_{\text{tmp}}$表示某项用户簇的偏好均值. 当$\overline{C}^{i}_{\text{tmp}}<\gamma$时, 即使$\beta\leq\hat{r}_{u,i}<\alpha$, 项目 i 也不会被推荐. 这因为当$\overline{C}^{i}_{\text{tmp}}<\gamma$时, 给对项目 i 进行惩罚. 当$\hat{r}_{u,i}>\beta$和$\overline{C}^{i}_{\text{tmp}}\geq\gamma$时, 将向用户 u 推荐项目 i (参考第 13、14 行). 当$\hat{r}_{u,i}<\beta$ (参考第 15行) 时, 项目 i 不会被推荐.

求解式 (5) 中的优化问题, 确定阈值 α、β 和 γ. 在迭代执行算法 2 的同时不断改变 α、β 和 γ 值. 也就是说, 根据式 (5), 迭代执行算法 2 中的第 4~17 行, 来获取 α、β 和 γ 的最优值.

使用 IPU 模型的 CBCF 算法总结如下:

(1) 通过使用 IPU 模型以及 CBCF 算法来决策是否将项目 i 推荐给活跃用户 u.

(2) 当 (即$\hat{r}_{u,i}\geq\alpha$), 那么将项目 i 推荐给用户 u.

(3) 当$\hat{r}_{u,i}\geq\beta$和$\overline{C}^{i}_{c}\geq\gamma$, 向用户 u 推荐项目 i, $\overline{C}^{i}_{c}$是用户簇C_c对项目 i 的偏好均值.

3 实验

3.1 实验数据集

本节描述数据集以及数据结构. CBCF 常用于非冷启动用户, 但它对冷启动用户同样有效. 本文使用 MovieLens 数据集下的载地址为: https://grouplens.org/datasets/movielens/, 其中有好几种版本, 对应不同数据量, 本文所用的数据为 100 KB 的数据集.

100 KB 的数据集具有以下属性:

(1) 评分最高为 5 星;

(2) 每个用户至少有 20 条评分记录;

(3) 100 KB 数据集有 100 000 条评分记录;

(4) 100 KB 数据集有 943 个用户和 1682 部电影.

值得注意的是, 从 movieens 100 KB 数据集获得的评分矩阵的稀疏度 (即评分矩阵中丢失的单元格数与单元格总数的比值) 为 93.7%. 数据稀疏性问题的一个普遍解决方法是采用数据填补的方式, 对缺失的单元格用零补全[18].

即使采用数据填补的方法能显著提高预测精度, 本文的重点在给定准确率的条件下最大化召回率 (或 F1- score) 而非解决数据稀疏性问题, 因此没有采用数据填补方法. 数据结构描述如下.

假设推荐系统中有一组用户 U 和一组项目 I, 如下:

$$\begin{cases} U\overset{\Delta}{=}\{u_1,u_2,\cdots,u_n\} \\ I\overset{\Delta}{=}\{i_1,i_2,\cdots,i_m\} \end{cases} \tag{7}$$

其中, n 和 m 分别表示用户数和项目数. 评分矩阵R_{CBCF}如下:

$$R_{\text{CBCF}}=\begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{1,m} \\ r_{2,1} & r_{2,2} & r_{2,3} & \cdots & r_{2,m} \\ r_{3,1} & r_{3,2} & r_{3,3} & \cdots & r_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & r_{n,3} & \cdots & r_{n,m} \end{pmatrix} \tag{8}$$

其中, $r_{u,i}$是用户 u 对项目 i 的评分, 其中$u\in\{1,\cdots,n\}$ $i\in\{1,\cdots,m\}$. 值得注意的是, R_{CBCF}可以是用户的显式评分, 也可以是用户的隐式评分. 如果用户 u 未对项目

i 进行评分, 那么$r_{u,i}$为空.

将用户集 U 分为多个用户簇, 用户簇是评分矩阵R_{CBCF}中的一组类似的用户. 为了聚类 U, 定义了 n 个用户向量, 每个用户向量由 m 个元素组成.

$$U_b = [r_{b,1}, r_{b,2}, \cdots, r_{b,m}] \tag{9}$$

对于$b \in \{1, \cdots, n\}$. 假设 n 个用户向量聚集到 C 个用户簇中, 其中的用户簇集合 C 表示为:

$$C = \{C_1, C_2, \cdots, C_c\} \tag{10}$$

同一个用户簇中用户的偏好相似度比不在同一个用户簇中的其他用户更接近. 例如, 假设有 4 个用户向量, 分别为 $u1$=[2, 0, 1, 0], $u2$=[0, 4, 0, 2], $u3$=[3, 0, 2, 0] 和 $u4$=[0, 3, 0, 2]. 需要将这 4 个向量分成 2 个簇. 那么, $u1$ 和 $u3$ 将被分到一个族中, 根据用户的评分, 他们是相似用户, 因为 ($u1$, $u3$) 之间的欧几里得距离比 ($u1$, $u2$)、($u1$, $u4$)、($u3$, $u2$) 和 ($u3$, $u4$) 等其他组合的欧几里得距离更近.

数据结构如表 2 所示. 数据由以下 3 个字段组成: 用户 ID、项目 ID 和用户评分. 例如, 如果用户 $u1$ 对项目 $i1$ 的评分为 4.0, 那么插入一个新的记录“$u1|i1|4.0$”.

表 2 数据结构

用户 ID	项目 ID	评分(R_{CBCF})
$u1$	$i1$	$r_{1,1}$
$u1$	$i2$	$r_{1,2}$
$u1$	$i8$	$r_{1,8}$
⋮	⋮	⋮
un	$i(m-4)$	$r_{n,m-4}$
un	im	$r_{n,m}$

3.2 实验结果与分析

本文从准确率、召回率和 F1-score 方面来评估使用 IPU 模型 CBCF 算法的性能. 本实验中, 除特殊说明外, 默认采用基于项目的 CF, 因为它在基于邻域 CF 推荐的准确率上有更好的性能, 这将在本节后面进行验证. 本文使用 Apache Mahout 来构建执行机器学习任务 (如 CF、聚类和分类) 的环境. 当满足以下条件时, 假设推荐结果为真:

(1) 实际评分为 4.0 或 5.0 的项目向用户推荐.

(2) 实际评分低于 4.0 的项目不向用户进行推荐.

本实验中, 谱聚类和 FCM 聚类算法均设置 c=10; 根据文献 [19] 将 FCM 聚类的模糊度 m 设置为 2; 并将 FCM 聚类的收敛阈值设置为10^{-4}. 在 FCM 聚类中, 将对象分配给具有最高系数的聚类. 本实验中, 除特殊说明外, 默认采用谱聚类. 图 2 比较了簇间和簇内的欧几里德距离, 以证明聚类的效果. PCC 的值在–1.0 和 1.0 之间, 其中 1.0 和–1.0 意味着两个对象 (如用户) 分别具有最高的正相关和负相关. 由于大多数聚类算法不采用负相关, 因此两个用户 $u1$ 和 $u2$ 之间的 PCC 值, 即 $s(u1,u2)$, 如下所示:

$$\begin{cases} s(u_1,u_2) \leftarrow 1 - s(u_1,u_2) \text{ for } s(u_1,u_2) \in [0,1] \\ s(u_1,u_2) \leftarrow -(s(u_1,u_2) - 1) \text{ for } s(u_1,u_2) \in [-1,0] \end{cases} \tag{11}$$

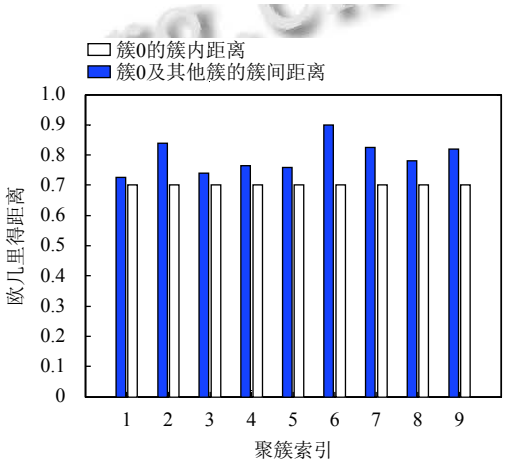

图 2 簇间和簇内的欧几里德距离比较结果

接近 0 表示高度正相关, 而接近 2 表示高度负相关. 如图 2 所示, 从簇 0 的角度看, 簇内距离小于簇间距离. 因此, 基于 PCC 的聚类是有效的.

图 3 显示了当阈值 γ 设置为 3.4, α 和 β (分别对应于给予惩罚和激励的阈值) 对 F1 分数的影响. 当 α=3.7 和 β=2.9 时, 使用 IPU 模型的 CBCF 算法的 F1 的最大值为 0.7451. 实验结果表明, 随着 α 和 β 的增加, F1-score 降低, 因为随着 α 和 β 的增加, 召回率的下降幅度大于准确率的上升幅度. 如果 α 和 β 都很大, 那么准确率和召回率分别增加和减少. 然而, 由于召回率的下降幅度大于准确率的上升幅度, F1-score 也相应降低. 例如, 在图 3 中, 当 α=3.7, β=2.9, γ=3.4 时, 准确率为 0.6595, 召回率为 0.8564; 当 α=4.4, β=4.4, γ=3.4 时, 准确率为 0.6853, 召回率为 0.076.

图 4 显示当采用不考虑聚类 (即 γ=0) 的基于项目 CF 算法时, F1-score 随着推荐阈值的变化趋势, 如果某个项目的预测评分大于推荐阈值, 那么将向用户推荐相应的项目. 如果实际评分超过 4.0, 那么该推荐是有效的. 如图 4 所示, 当阈值为 3.1 时, F1-score 的最大值

为 0.7282. 实验结果表明, 总体趋势与图 3 相似, 与不采用聚类的基于项目的 CF 的算法相比, 本文所提出算法的 F1-score 提高近 3%.

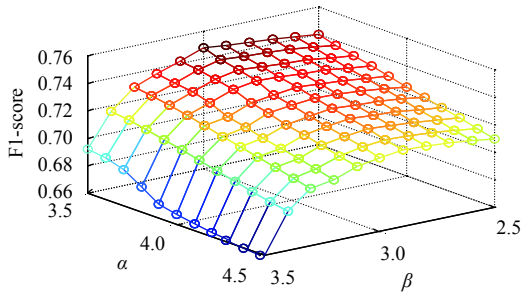

图 3 当阈值 γ 为 3.4, α 和 β 对 F1-score 的影响

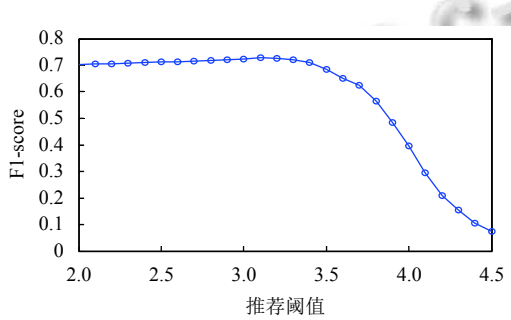

图 4 基于项目的 CF, F1-score 随着推荐阈值的变化趋势

表 3 显示了使用 IPU 模型 CBCF 算法和不使用聚类的基于项目的 CF 算法在给定准确率情况的召回率和 F1-score. 基于项目的 CF 算法 (未使用聚类算法) 中, 当阈值设置为 4.0, 准确率为 0.7449 时, 召回率的最大值为 0.2815. 使用 IPU 模型 CBCF 算法中, 当准确率为 0.7449, α=3.9, β=2.1, γ=4.2 时, 召回的最大值为 0.4343. 召回率提高近 50%. 也就是说, 如表 3 所示, 在准确率相同的情况下, 与基于项目的 CF(未使用聚类算法) 相比, 本文提出的算法具有非常高的召回率. 从图 3、图 4 和表 3 可见, 本文提出的算法在给定准确率的情况下, 召回率或 F1-score 得到了很大的改进.

表 3 给定准确率情况的最大召回率和 F1-score.

准确率	不使用聚类的CF算法		使用IPU模型的CBCF算法	
	召回率	F1-score	召回率	F1 score
0.7449	0.2815	0.4085	0.4343	0.5487
0.7201	0.4565	0.5588	0.5706	0.6367
0.7074	0.5499	0.6188	0.6842	0.6956
0.6519	0.7914	0.7149	0.8251	0.7283
0.6036	0.9177	0.7282	0.9402	0.7352

一般来说, 推荐阈值越小, 准确率越低, 召回率越高, 反之亦然. 然而, 如前所述, 当阈值变得非常大时, F1-score 会迅速下降, 因为召回率的下降幅度大于准确率的上升幅度.

本文提出的 CBCF 算法中, 也可以使用基于用户的 CF 代替基于项目的 CF 通过求解式 (5), 找到参数 α, β 和 γ 最优值, 对使用 IPU 模型基于项目的 CF 算法分别使用谱聚类和 FCM 聚类算法的性能进行比较, 如表 4 所示, 以及对采用 IPU 模型基于用户的 CF 算法分别使用谱聚类和 FCM 聚类算法的性能进行比较, 如表 5 所示, 以上测试数据都不包含冷启动用户. 根据实验结果, 总结如下: 1) 基于项目的 CF 算法比基于用户的 CF 在 F1 分数上有更好的表现; 2) 基于 FCM 聚类的算法比谱聚类有更好的表现.

表 4 比较本文提出的算法 (基于项目的 CF) 分别使用谱聚类和 FCM 聚类算法的性能

聚类	γ	α	β	准确率	召回率	F1-score
谱聚类	3.4	3.7	2.9	0.6595	0.8564	0.7451
FCM	3.5	3.3	2.5	0.6625	0.8639	**0.7499**

表 5 比较本文提出的算法 (基于用户的 CF) 分别使用谱聚类和 FCM 聚类算法的性能

聚类	γ	α	β	准确率	召回率	F1-score
谱聚类	3.5	3.1	2.7	0.6309	0.8893	0.7382
FCM	3.3	3.7	2.9	0.6448	0.8730	**0.7418**

此外, 还比较了本文提出的算法 (采用谱聚类) 与未使用聚类的 CF 算法基于冷启动用户 (评分项目数少于 20) 数据的性能. 由于 movieens 100 KB 数据集不包含冷启动用户的记录, 根据文献 [20] 修改了实验设置. 具体来说, 本文选取了有过 20~30 部电影评分记录的 290 名用户的数据作为测试集, 并随机抽取每个用户 3~20 个评分项目. 原始数据集中剩余的 653 个用户作为训练集. 表 6 中的结果与非冷启动用户的结果具有相似的趋势, CBCF 算法的性能优于未采用聚类的 CF 算法.

表 6 比较本文提出的算法与未采用聚类的 CF 算法基于冷启动用户数据的性能

算法	准确率	召回率	F1-score
未采用聚类的CF算法	0.7085	0.3552	0.4732
本文提出的算法	0.6793	0.6934	**0.6863**

4 结束语

通过对推荐系统的不断探索, 本文提出了使用

IPU 模型的 CBCF 算法, 并提出了一个约束优化问题, 即在给定准确率的条件下最大化召回率 (或 F1-score). 为此, 应用聚类算法, 根据实际评分和皮尔逊相关系数将用户分为多个聚类, 并根据同一个聚类内用户的偏好倾向, 对每个项目进行激励或惩罚. 实验结果表明, 采用 IPU 模型的 CBCF 算法在给定准确率的条件下, 召回率或 F1-score 有显著地提高. 本文未来研究的一个方向是通过利用基于模型的 CF 算法 (如矩阵分解) 的特性, 设计一种新的基于聚类的 CF 算法.

参考文献

1 姜书浩, 张立毅, 张志鑫. 基于个性化的多样性优化推荐算法. 天津大学学报 (自然科学与工程技术版), 2018, 51(10): 1042–1049.

2 Su XY, Khoshgoftaar TM. A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009: 421425.

3 朱叶. 面向海量数据的推荐系统的研究 [硕士学位论文]. 北京: 北京理工大学, 2015.

4 Cai Y, Leung HF, Li Q, *et al*. Typicality-based collaborative filtering recommendation. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(3): 766–779. [doi: 10.1109/TKDE.2013.7]

5 翁小兰, 王志坚. 协同过滤推荐算法研究进展. 计算机工程与应用, 2018, 54(1): 25–31.

6 Sobhanam H, Mariappan AK. A hybrid approach to solve cold start problem in recommender systems using association rules and clustering technique. International Journal of Computer Applications, 2013, 74(4): 17–23. [doi: 10.5120/12873-9697]

7 Andrat H, Ansari N. Analyzing game stickiness using clustering techniques. In: Bhatia SK, Mishra KK, Tiwari S, *et al*, eds. Advances in Computer and Computational Sciences. Singapore: Springer, 2018. 645–654.

8 Chowdhury K, Chaudhuri D, Pal AK. A novel objective function based clustering with optimal number of clusters. In: Mandal JK, Mukhopadhyay S, Dutta P, *et al*, eds. Methodologies and Application Issues of Contemporary Computing Framework. Singapore: Springer, 2018. 23–32.

9 Tehreem A, Khawaja SG, Khan AM, *et al*. Multiprocessor architecture for real-time applications using mean shift clustering. Journal of Real-Time Image Processing, 2019, 16(6): 2233–2246. [doi: 10.1007/s11554-017-0733-0]

10 Huang CL, Yeh PH, Lin CW, *et al*. Utilizing user tag-based interests in recommender systems for social resource sharing websites. Knowledge-Based Systems, 2014, 56: 86–96. [doi: 10.1016/j.knosys.2013.11.001]

11 Yin B, Yang YJ, Liu WH. Exploring social activeness and dynamic interest in community-based recommender system. Proceedings of the 23rd International Conference on World Wide Web. Seoul, Republic of Korea. 2014. 771–776.

12 Koohi H, Kiani K. User based collaborative filtering using fuzzy C-means. Measurement, 2016, 91: 134–139. [doi: 10.1016/j.measurement.2016.05.058]

13 Wu Y, DuBois C, Zheng AX, *et al*. Collaborative denoising auto-encoders for top-N recommender systems. Proceedings of the 9th ACM International Conference on Web Search and Data Mining. New York, NY, USA. 2016. 153–162.

14 Yang Z, Wu B, Zheng K, *et al*. A survey of collaborative filtering-based recommender systems for mobile Internet applications. IEEE Access, 2016, 4: 3273–3287. [doi: 10.1109/ACCESS.2016.2573314]

15 von Luxburg U. A tutorial on spectral clustering. Statistics and Computing, 2007, 17(4): 395–416. [doi: 10.1007/s11222-007-9033-z]

16 闫岩. 一种基于情境聚类的协同过滤算法的研究与实现. [硕士学位论文]. 长沙: 湖南大学, 2016.

17 Xu T, Tian J, Murata T. Research on personalized recommendation in E-commerce service based on data mining. Proceedings of International MultiConference of Engineers and Computer Scientists. Hong Kong, China. 2013. 313–317.

18 Hwang WS, Parc J, Kim SW, *et al*. "Told you I didn't like it": Exploiting uninteresting items for effective collaborative filtering. Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering. Helsinki, Finland. 2016. 349–360.

19 Pal NR, Bezdek JC. On cluster validity for the fuzzy c-means model. IEEE Transactions on Fuzzy Systems, 1995, 3(3): 370–379. [doi: 10.1109/91.413225]

20 Jazayeriy H, Mohammadi S, Shamshirband S. A fast recommender system for cold user using categorized items. Mathematical and Computational Applications, 2018, 23(1): 1. [doi: 10.3390/mca23010001]