

基于混合蛙跳联合聚类的协同过滤算法

喻金平, 张 勇, 廖列法, 梅宏标

(江西理工大学 信息工程学院, 江西 赣州 341000)

摘 要: 针对传统协同过滤推荐算法稀疏性、冷启动、推荐质量不高等缺陷, 提出一种基于混合蛙跳模糊聚类的协同过滤推荐算法。该算法先对原始评分矩阵用户和项目进行联合聚类, 利用联合聚类结果对评分矩阵进行填充, 再对混合蛙跳算法进行改进, 利用改进后的算法快速地全局寻优能力得到项目最近邻居集合, 最后通过计算预测评分生成推荐结果。仿真结果表明, 该算法有效缓解对评分数据稀疏性的不良影响, 同时在推荐精度上有明显改善。

关键词: 推荐系统; 协同过滤; 联合聚类; 数据填充; 混合蛙跳

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2016)01-0065-07

Collaborative Filtering Recommendation Based on Shuffled Frog Leaping Fuzzy Co-Clustering Algorithm

YU Jin-ping, ZHANG Yong, LIAO Lie-fa, MEI Hong-biao

(School of Information Engineering, Jiangxi University of Science & Technology, Ganzhou 341000, China)

Abstract: In order to overcome the disadvantages of the traditional collaborative filtering recommendation algorithm, such as sparsity, cold start and low recommend quality, collaborative filtering recommendation based on shuffled frog leaping fuzzy co-clustering algorithm was proposed. First, co-clustering algorithm is used to simultaneously obtain user and item neighborhoods for the original score matrix, and then the results of co-clustering is used on rating matrix. Improve the shuffled frog leaping fuzzy, and the improved-shuffled frog leaping algorithm is used based on its fast global optimization ability to get the nearest neighbor set. Lastly, the final rating prediction is obtained. The experimental result show that filtering recommendation based on shuffled frog leaping fuzzy co-clustering algorithm will become more accurate, which can effectively relieve the impact of sparse data and improve the quality of recommendation.

Key words: recommendation systems; collaborate filtering; co-clustering; data padding; shuffled frog leaping fuzzy

1 引言

随着互联网的快速发展, 人们正处在一个信息迅速增长的时代, 搜索工具的出现, 一定程度上解决了人们在信息筛选问题。但是大部分用户还是不能准确的描述自己的信息需求或者不会使用搜索工具, 在这样的背景下, 推荐系统的出现解决了这种问题, 一边帮助用户接收到对自己有用的话题, 另一边帮助商家快速的把信息投放到感兴趣的人群中。

协同过滤 (Collaborative Filtering, CF) 推荐算

法是目前推荐算法中最成功推荐算法之一, 目前已经被应用于各个领域, 并取得了很不错的效果。但是传统的 CF 算法^[1]存在且数据的稀疏性和冷启动问题, 推荐的质量不高, 针对以上问题, 查文琴等^[2]给出了一种改进用户聚类的 CF 方法, 从用户评价和用户对项目关注这两个方面考虑对推荐的影响; 邵琳琳^[3]提出了基于混合蛙跳模糊聚类的 CF 算法, 利用模糊聚类对数据进行预处理, 然后利用混合蛙跳算法找到用户最近邻居集合, 提高了推荐的精度。韦素云等^[4]提出了基于联合聚类平滑的协同

过滤算法,利用联合聚类平滑填充的方法去解决数据稀疏性问题;除了使用聚类的方法,李海霞等^[5]将仿生算法应用到推荐系统中,为 CF 算法的改进提出了新的研究方法.然而,以上的这些方法都不能全面的去解决传统协同过滤算法存在的问题,本文提出了一种基于混合蛙跳联合聚类的协同过滤算法.

2 相关算法

2.1 蛙跳算法

混合蛙跳算法 (Shuffled frog leaping algorithm, SFLA) 最早于 2003 年提出来的. SFLA 算法的基本原理如下^[6]:

(1) 在 $\Omega \subset R^d$ 中,随机初始化 f 只青蛙,组成 $x(1), x(2), \dots, x(f)$ 种群,初始种群中每一只青蛙当前的位置表示定义域内的一个备选的解 $X = (x_{i1}, x_{i2}, \dots, x_{id})$,解向量中的 d 为维变量;

(2) $f(X_i)$ 表示初始种群中第 i 只青蛙个体的适应值,计算初始种群的初始适应度值,将 f 只青蛙指定适应度值按照降序进行排列,并找出适应度值最好的解 X_g ;

(3) 将所有的蛙群分成 m 个子群组,每个模因组当中含有了 n 只青蛙.在迭代过程中,第 1 只青蛙进入 M_1 子群组中,第 2 只青蛙进入 M_2 子群组中,一直到第 M 只青蛙分到了 M_m 子群组中,之后,将第 $M+1$ 只青蛙进入 M_1 子群组中,第 $M+2$ 只青蛙进入 M_2 子群组中,以此类推,第 $2M$ 只青蛙分到了 M_m 子群组中,直到所有的青蛙都分配完为止.

局部搜索:每个子群组中,用 P_b 和 P_w 表示该子群组中位置最好的青蛙和位置最差的青蛙,在所有的青蛙群体中,位置最好的青蛙记做为 P_g .在进化过程中,对每个子群组中的青蛙的最差位置 P_w 进行更新操作,更新公式如下:

$$\begin{cases} E_j = \text{rand}() \times (P_b - P_w), j = 1, 2, \dots, d \\ \text{new}P_w = \text{old}P_w + E_j - E_{\max} \leq E_j \leq E_{\max} \end{cases} \quad (1)$$

式中,分量在 j 上移动的距离为 E_j ,青蛙个体所移动的最大步长为 $E_{\max}$,按照上面的过程将 P_w 更新为 $\text{new}P_w$,如果更新后的 $\text{new}P_w$ 相对于之前 P_w 有改进,也就是青蛙个体的位置变的更好,则之前的位置最好的青蛙 P_w 会被 $\text{new}P_w$ 替代.如果仍然没有得到相应的改进,那么在定义域内以同等概率生成一个新位置取代之前的 P_w ,在算法规定的进化次数内重复进行以上的更新过程,这就是 SFLA 的一次局部搜索过程.

$$\begin{cases} E_j = \text{rand}() \times (P_g - P_w), j = 1, 2, \dots, d \\ \text{new}P_w = \text{old}P_w + E_j - E_{\max} \leq E_j \leq E_{\max} \end{cases} \quad (2)$$

全局信息交换:经过算法规定的进化次数的局部搜索过程后,将所有的子群的青蛙个体 ($m \times n$) 全部放在一起,指定适应度值按照降序进行排列,并且更新整个群体中的青蛙最好位置 P_g ,再次划分青蛙子群,继续执行局部搜索的过程,直到符合指定的条件停止算法操作.

2.2 CF 算法

CF 算法,主要包括 3 个部分:项目的数据、相似性的度量、预测评分^[7].

(1) 数据描述

在推荐系统中,定义数据 $D = (U, P, R)$,其中, $U = \{u_1, u_2, \dots, u_m\}$ 是 m 用户集合, $P = \{p_1, p_2, \dots, p_n\}$ 是 n 个项目的集合, R 是用户对各项目评分矩阵.

(2) 相似性度量

常见的相似性度量方法有:余弦相似性、修正余弦相似性、Pearson 相关相似性,相关相似性是目前最常用的度量方法,因此,本文采用该方法进行相似性度量.设项目 j, j' 相似性 $\text{sim}(j, j')$ 为:

$$\text{sim}(j, j') = \frac{\sum_{i \in I_{u,v}} (r_{ij} - \bar{r}_j)(r_{ij'} - \bar{r}_{j'})}{\sqrt{\sum_{i \in I_{u,v}} (r_{ij} - \bar{r}_j)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{ij'} - \bar{r}_{j'})^2}} \quad (3)$$

式中, r_{ij} 是用户 i 对项目 j 的评分; $\bar{r}_j$ 是在已评分项目上的平均值.

(3) 预测评分

对于项目 j ,计算其与其他项目的相似性,相似性最大的 k 个项目组成项目 j 的最近邻居集为 N_j ,则用户 a 对项目 j 的预测评分 $A_{a,j}$ 为:

$$A_{a,j} = \bar{r}_a + \frac{\sum_{u \in S_U^j} \sin(j, j')(r_{a,j'} - \bar{r}_j)}{\sum_{u \in S_U^j} |\sin(j, j')|} \quad (4)$$

式中, $\bar{r}_a$ 和 $\bar{r}_j$ 分别是用户 a 和项目 j 的平均评分值.

2.3 联合聚类

联合聚类 (Co-Clustering)^[8] 指的是对数据矩阵的行和列进行双向聚类,是聚类方法的一种,是目前一种非常常用的数据分析的技术,相对于传统的聚类,Co-Clustering 更容易寻找隐藏在数据中的信息. Co-Clustering 在基因组分析、推荐系统中的数据分析、自然语言处理与图像语音及视频分析、用户

网页浏览历史数据分析等大量应用领域取得具大成功。

基于信息论的联合聚类算法^[9]是指利用最优的 Co-Clustering 使聚类前后的互信息熵损失达到最小。假设有 m 个用户,记作 $\{i_1, i_2, \dots, i_m\}$; 有 n 个项目,记作 $\{j_1, j_2, \dots, j_m\}$; 那么用 $p(I, J)$ 表示变量 I 和 J 的联合概率分布,这种矩阵叫做二维列联表。假设待聚类的 m 个用户属于 k 个类,记作 $\{i_1^*, i_2^*, \dots, i_k^*\}$; 待聚类的 n 个项目属于 l 个类,记作 $\{j_1^*, j_2^*, \dots, j_l^*\}$,联合聚类算法的目的就是找到使得 $\{i_1, i_2, \dots, i_m\}$ 到 $\{i_1^*, i_2^*, \dots, i_k^*\}$ 的映射, $\{j_1, j_2, \dots, j_m\}$ 到 $\{j_1^*, j_2^*, \dots, j_l^*\}$ 的映射,传统的算法只是寻找其中一个映射。

在联合聚类中引入信息熵的理论,定义为 $I(I, J)$,代表变量 I 和 J 之间公有的信息量,为了使得聚类前后的互信息熵损失最小,即要满足:

$$\operatorname{argmin}\{I(I, J) - I(I^*, J^*)\} \quad (5)$$

3 基于混合蛙跳联合聚类的协同过滤算法

3.1 基于用户和项目的联合聚类

推荐算法主要是指预测感兴趣的用户对未知评分项目的评分。具体描述如下:假设已知数据 $D = (U, P, R)$,那么推荐算法的作用就是预测评分矩阵 R 中的未知评分(也就是 R 中空缺的位置)。由此可以将推荐问题转化成利用 Co-Clustering 方法求逼近矩阵问题:首先定义逼近矩阵 $\hat{R}$; 然后利用 Co-Clustering 方法找到用户和项目的联合聚类 (α, β) ,使得评分矩阵 R 和逼近矩阵 $\hat{R}$ 之间的误差值达到一定的阈值(阈值是自定义的)。评分矩阵 R 和逼近矩阵 $\hat{R}$ 之间的误差值可以由下式表示:

$$\min_{(\alpha, \beta)} \sum_{i=1}^m \sum_{j=1}^n w_{ij} (r_{ij} - \hat{r}_{ij})^2 \quad (6)$$

式中, w_{ij} 是指评分标识矩阵 W 中的元素,当用户 i 对项目 j 已经做出评分,即 $r_{ij} \neq 0$ 时,矩阵 W 中的元素 $w_{ij} = 1$, 否则 $w_{ij} = 0$ 。 $\hat{R}$ 中的元素 $\hat{r}_{ij}$ 不仅考虑 Co-Clustering 的均值,同时还考虑用户和项目的偏好,即:

$$\hat{r}_{ij} = \bar{r}_{\alpha(i)\beta(j)} + (\bar{r}_i - \bar{r}_{\alpha(i)}) + (\bar{r}_j - \bar{r}_{\beta(j)}) \quad (7)$$

式中, $\bar{r}_i$ 和 $\bar{r}_j$ 是用户 i 和项目 j 的平均评分值, $\bar{r}_{\alpha(i)\beta(j)}$, $\bar{r}_{\alpha(i)}$ 和 $\bar{r}_{\beta(j)}$ 分别是 Co-Clustering 的平均评分值、用户聚类平均评分值和项目聚类平均评分值,即:

$$\begin{aligned} \bar{r}_{\beta(j)} &= \frac{\sum_{i'=1}^m \sum_{j' \in \beta(j^*)} r_{i'j'}}{\sum_{i'=1}^m \sum_{j' \in \beta(j^*)} w_{i'j'}}, \bar{r}_j = \frac{\sum_{i'=1}^m r_{i'j}}{\sum_{i'=1}^m w_{i'j}}, \\ \bar{r}_{\alpha(i)} &= \frac{\sum_{j'=1}^n \sum_{i' \in \alpha(i^*)} r_{i'j'}}{\sum_{j'=1}^n \sum_{i' \in \alpha(i^*)} w_{i'j'}}, \bar{r}_{\beta(j)} = \frac{\sum_{i'=1}^m \sum_{j' \in \beta(j^*)} r_{i'j'}}{\sum_{i'=1}^m \sum_{j' \in \beta(j^*)} w_{i'j'}} \end{aligned}$$

Co-Clustering 算法的具体步骤如下:

(1) 初始化,同等概率下划分矩阵的行和列,产生用户和项目的联合聚类 (α, β) ;

(2) 计算 Co-Clustering 的各个均值, $\bar{r}_{\beta(j)}$, $\bar{r}_i$, $\bar{r}_j$, $\bar{r}_{\alpha(i)}$ 和 $\bar{r}_{\beta(j)}$;

(3) 新行聚类:

$$\alpha(i) = \operatorname{argmin}_{i=1}^m \sum_{j=1}^n w_{ij} (r_{ij} - \bar{r}_{\alpha(i)\beta(j)} - \bar{r}_i + \bar{r}_{\alpha(i)} - \bar{r}_j + \bar{r}_{\beta(j)})^2, 1 \leq i \leq m;$$

(4) 更新列聚类:

$$\beta(j) = \operatorname{argmin}_{j=1}^n \sum_{i=1}^m w_{ij} (r_{ij} - \bar{r}_{\alpha(i)\beta(j)} - \bar{r}_i + \bar{r}_{\alpha(i)} - \bar{r}_j + \bar{r}_{\beta(j)})^2, 1 \leq j \leq n;$$

(5) 终止条件,计算式(6),如果计算的结果小于预定义的值,则停止迭代,输出结果;否则返回(2),继续运行。

下面分别从用户聚类簇、项目聚类簇和联合聚类簇三个方面对 R 进行填充,过程如下:

(1) 若 $i \in U, j \in P$, 则 $r_{ij} = \bar{r}_i + \bar{r}_j - \bar{r}_{\alpha(i)} - \bar{r}_{\beta(j)} + \bar{r}_{\alpha(i)\beta(j)}$;

(2) 若 $i \in U, j \notin P$, 则 $r_{ij} = \bar{r}_i$;

(3) 若 $i \notin U, j \in P$, 则 $r_{ij} = \bar{r}_j$;

(4) 若 $i \notin U, j \notin P$, 则 $r_{ij} = \bar{R}$ 。

3.2 改进的 SFLA 求解最近邻居集

3.2.1 改进的 SFLA

混合蛙跳算法思想简单,执行过程中调整的因子很少,相对于其他的智能算法,在全局信息交流的能力较强。但是同时也有搜索速度缓慢、局部信息和全局信息交流不全面等缺点,针对以上缺点,本文在种群初始化、个体变异、种群分组三个方面对蛙跳算法进行改进。

(1) 对种群初始化的改进

传统混合蛙跳算法在种群初始化时,采用随

机的方法产生,这种初始化方法具有很大的不稳定性,如果初始值接近于最优解,则算法的收敛速度加快,如果初始值不接近于最优解,则收敛速度很慢,针对这种情况,本文参照反向差分进化的思想对蛙跳算法进行初始化种群.首先了解一个基本概念^[10]:

定义 1 若 $X = (x_1, x_2, \dots, x_n)$ 为 N 空间中的一个点, $x_i \in [a_i, b_i], i = 1, 2, \dots, N$, 则 X 的反向点为 $\hat{X} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$, 那么如图 1 所示.

$$\hat{x}_i = a_i + b_i - x_i \quad (8)$$

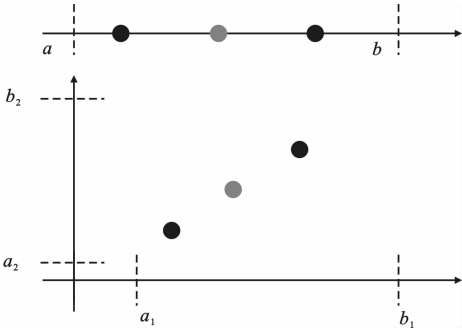


图 1 反向点示意图

根据文献[7]可知,反向值的有更大的可能接近于函数的最优值.则改进后的混合蛙跳算法的初始化种群过程可以描述如下:

步骤 1: 在 N 维空间中,随机产生初始解 $X = [x_1, x_2, \dots, x_n]$;

步骤 2: 利用式(8)计算 X 的反向解 $\hat{X} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n]$;

步骤 3: 计算上述随机解和反向解的适应度函数,若 $f(X) > f(\hat{X})$, 则选择 X , 若 $f(X) < f(\hat{X})$, 则选择 $\hat{X}$.

(2) 青蛙个体变异方式的改进

在 PSO 中,首先初始化粒子群,然后粒子们就跟随当前的最优粒子在解空间中搜索.通过迭代找到最优解,迭代粒子公式如下^[11]:

$$\begin{cases} v(i) = \omega \times v(i) + c_1 \times \text{rand}() \times [P_b(i) - P(i)] \\ \quad + c_2 \times \text{rand}() \times [P_g(i) - P(i)] \\ \text{new}P(i) = \text{old}P(i) + v(i) \end{cases} \quad (9)$$

式中, $v(i)$ 是粒子当前的速度, ω 是惯性系数, $\text{rand}()$ 是 0 ~ 1 之间的随机数, P_b 是个体极值, P_g 是全局极值, $p(i)$ 是粒子的当前位置, c_1 和 c_2 是学习因子.

蛙跳算法子群在进行局部搜索时,更新 P_w 此时利用的只是局部最优的个体信息,在全局搜索时,

更新 P_g 利用的只是全局最优的个体信息,使得个体信息和子群信息之间无法交流,从而不能得到最优个体信息;每个子群进行局部深度搜索时,如果更新 $\text{new}P_w$ 不优于之前最好位置 P_w , 则随机产生一个 P_w , 这样就失去了之前的全部信息.因此,我们可以利用粒子群优化个体极值和全局极值的思想,在蛙跳算法的进化过程中加入个体最优值和全局最优值,保证了蛙跳算法在局部搜索和全局搜索时保证了局部最优信息和全局最优信息的交流,分量 j 借鉴粒子群优化思想引入上一次移动距离 E_{j-1} 的惯性权重系数来调节移动距离,使算法在进化过程中,对延续上一次更新的信息的保留,提高算法的寻优能力.同时文献[7]的研究也表明加入学习因子 c 也增加了算法的搜索速度,那么蛙跳算法的更新公式为:

$$E_j = \omega E_{j-1} + \text{rand}() \times (P_b - P_w) \quad (10)$$

其中 $j = 1, 2, \dots, d$. 原始的混合蛙跳算法在进行个体更新时,考虑的是最优和最差青蛙个体之间的步长差,然而子群内青蛙的平均值比最优青蛙位置更能代表子群内青蛙位置.因此利用种群中的最优位置和历史最优位置之间的随机点出发,以子群内的青蛙的平均值和最差位置青蛙的差值为步长进行青蛙个体的更新操作,双向随机查找更优的青蛙.子群内的青蛙的平均值为 $\bar{P}$, 令 $r = \text{rand}()$, 更行公式为:

$$\begin{cases} E_j = \omega E_{j-1} + r \times (\bar{P} - P_w) \\ \text{new}P_w = [r \times P_g + (1-r)P_b] + E_j \end{cases} \quad (11)$$

如果更新后的 P_w 优于原来的 P_w , 则用新 P_w 更新原来的 P_w , 若新 P_w 不优于原来的 P_w , 则利用公式进行更新操作,若此时更新的 P_w 优于原来的 P_w , 则进行更新操作,如果不优于原来的,则随机生成 P_w , 再利用第 3.2.1 节中的方式选取 P_w 的反向值 $\hat{P}_w$, 进行下一轮迭代替换.

$$\begin{cases} E_j = \omega E_{j-1} + r \times (P_g - P_w) \\ \text{new}P_w = [r \times P_g + (1-r)P_b] + E_j \end{cases} \quad (12)$$

(3) 种群分组方法的改进

SFLA 在蛙群进行分组时,适应度较差的青蛙个体总是会被分在最后几组,使得处在后几组青蛙子群中最差个体向最好个体学习的效果不如处在前几组青蛙子群的好,这样导致局部搜索时有一定的局限性,容易陷入局部最优的可能^[12]. 针对于这种情况,本文对原始蛙跳算法的分组方法进行改进,开始的时候按照原始蛙跳算法的方法进行操作分成 m 组,每组含有 n 个青蛙个体,然后在在其他组选择一

个最优个体和随机选择一个青蛙个体放入本组,这样处理后,每组的青蛙个体个数变成 $n+m-1$,等待每组进化完成后,混合整个种群,此时,种群中的所有青蛙的个数为:

$$m(n+m-1) = m^2 + nm + m \quad (13)$$

再重新计算这些青蛙个体的适应度,取前 f 个个体进行下一次的迭代更新操作。

3.2.2 SFLA 相关元素表示

利用 SFLA 解决优化问题的主要有两个重要的步骤:青蛙粒子的编码和适应度函数的确定。同样,将 SFLA 应用于 CF 算法的邻居集的求解问题也要解决这两个问题:其一是如何将青蛙粒子编码;其二是如何构造适应值函数。

(1) 青蛙粒子编码:随机生成 f 只青蛙初始群体, $P = \{P_1, \dots, P_k, \dots, P_F\}$, 第 k 只青蛙可以描述为: $P_k = \{x_{kn} \mid n = 1, 2, \dots, w\}$ 。为了使得 SFLA 能够应用于协同过滤算法推荐中,本文对青蛙编码做特殊处理,即 $x_{kn} = 0$ 或 $x_{kn} = 1$,并且编码中为“1”的位置数恰好为 N 。

(2) 适应度函数:适应度函数是项目 j 与 j' 相似度值,即:

$$f(U) = \sum_{n=1}^w x_{kn} \text{sim}(j, j') \quad (14)$$

3.2.3 基于改进的 SFLA 的最近邻居集求解

SFLA 求解用户最近邻居集的基本思想为:将候选邻居集最为 SFLA 中青蛙粒子的位置,通过子群内的局部搜索和种群内全局信息的交换迭代,得到最近邻居集合。当完成用户和项目的双向聚类后,对于目标项目 j ,计算其与每个项目的聚类中心的相似度,选取前 l 个聚类作为候选邻居集合 $p' = \{j_1, j_2, \dots, j_w\}$,利用 SFLA 在 p' 中选取 N 个用户组成最近邻居集合。

基于 SFLA 的最近邻居集求解可以描述为:

步骤 1:初始化参数,聚类数为 l ,青蛙总数为 f ,子群数目为 m ,每个子群中有 n 只青蛙,初始的位移距离 E_j ,子群的最大迭代次数为 P ,全局的最大迭代次数为 Q ;

步骤 2:对群体中的每个青蛙粒子按式(14)计算适应度函数 $f[P_k]$,并将式(14)计算的值按照降序排列,同时找到全局最好的青蛙位置(最好解) P_g ;

步骤 3:确定每个子群里的青蛙的最优位置(最优解) P_b 和最差位置(最差解) P_w ;

步骤 4:对每个子群进行局部搜索,按照式(11)

计算位移距离 E_j ,如果 $\text{new}P_w$ 好于 $\text{old}P_w$,则更新 P_w ,如果不好于,则随机产生一个 P_w ,重新计算各只适应值进行降序排列,如果局部最优位置 P_b 优于全局最优位置 P_g ,则更新全局最优位置 P_g ,重复执行此过程达大子群最大迭代次数 P ,进行下一步;

步骤 5:全局信息交换,对已经进行更新操作的子群进行混合,利用第 3.2.1 中的改进种群分类对新种群进行分类,计算适应值函数 $f[P_k]$ 进行降序排列;

步骤 6:如果达到开始设置的全局最大迭代次数 Q ,则输出最优解,算法结束;否则,转向步骤 3 继续执行。

通过 SFLA 得到了项目的最近邻居集合。

3.3 基于改进混合蛙跳联合聚类的协同过滤算法

本文主要是将改进的混合蛙跳算法、联合聚类应用到传统的协同过滤算法中。首先,利用联合聚类从用户和项目进行双向聚类,对评分数据进行填充,然后再利用改进的混合蛙跳算法搜寻目标项目的最近邻居项目集合,最后利用求的最近邻居项目来预测估计其对未评分项目的评分,选择预测估计评分最高的前 N 项作为推荐结果推送给用户。

步骤 1:用户和项目联合聚类,扫描评分矩阵 R ,创建辅助矩阵 W ,得到生成用户和项目联合聚类及各个聚类簇均值;

步骤 2:利用联合聚类结果,对用户、项目评分矩阵 R 中未评分元素进行填充;

步骤 3:计算相似性,对于任意不同项目 j 和 j' 利用式(3)计算项目相似性 $\text{sim}(j, j')$;

步骤 4:对于每个项目 j ,利用改进后的混合蛙跳算法寻找最近邻居集 N_j ;

步骤 5:产生推荐,利用式(4)计算待推荐用户 a 对没有评分项目 j 的预测评分 $A_{a,j}$,取评分最高的前 N 个值所对应的项目组成推荐集 top-N 。

4 实验分析

本文实验所用算法由 Matlab 7.10.0 编程实现,本实验所使用计算机的 CPU 为 Intel Core i5、主频为 2.00 GHz、内存为 4 GB、操作系统为 Windows 8.1。

4.1 改进的 SFLA 性能测试

仿生算法的性能测试,最常见方法就是函数的优化。为了验证改进混合蛙跳算法的性能,选择两个标准测试函数进行实验,其中 f_1 为单峰函数, f_2 为多峰函数,标准函数的定义如表 1 所示。

表 1 标准函数表达式

函数名	表达式
$f_1(x)$	$f_1(x) = \sum_{i=1}^n x_i^2$
$f_2(x)$	$f_2(x) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2)^{0.25} [\sin(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1]$

青蛙的种群分组数 m 为 20, 每组青蛙包含的个数 n 为 10, 组内的迭代数 Ne 为 50, 最大步长 $S_{\max}$ 为 100, 种群的进化代数 MAXGEN 为 400. 实验结果如图 2、图 3 所示.

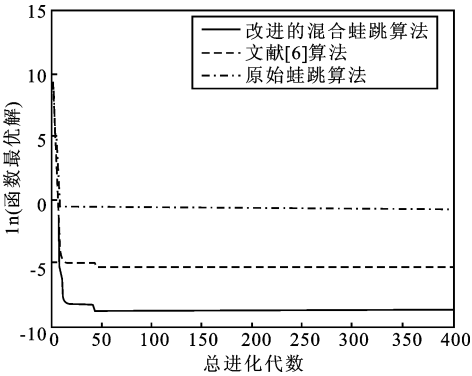


图 2 $f(1)$ 函数

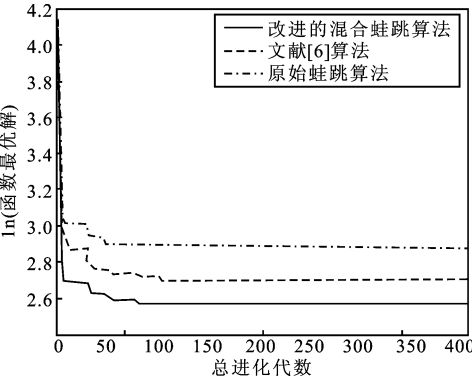


图 3 $f(2)$ 函数

根据图 2 和图 3 可知, SFLA 算法在标准测试函数上收敛速度相对于文献[6]和本文算法较为缓慢、容易陷入局部最优解的情况; 文献[6]算法引进了加速因子 c , 虽然相对原始算法在收敛速度有所提高, 但是还是容易过早的陷入局部最优; 本文的算法将通过对算法的初始化、个体更新迭代和种群分组三个方面进行改进, 加快算法的搜索速度和收敛速度, 同时克服了算法过早陷入局部最优.

4.2 基于混合蛙跳联合聚类协同过滤算法性能测试

4.2.1 数据集和评估标准

实验以公开的 MovieLens 站点所提供的数据

集(<http://movielens.umn.edu>)为实验环境. 该数据集中用户的数量为 943 个, 项目(电影)的数量为 1 682 个, 用户对电影的评分大约 10 万余条, 评分范围为 1~5.

评价一个推荐系统的质量主要采用的是平均绝对误差 MAE, 该方法主要是通过计算预测用户评分与实际的用户评分之间的偏差来度量预测的准确性. 假设用户的预测评分集合为 $\{P_1, P_2, \dots, P_n\}$, 对应的实际评分集合为 $\{R_1, R_2, \dots, R_n\}$, 则可由下式计算 MAE:

$$MAE = \frac{\sum_{i=1}^n |P_i - R_i|}{n}$$

4.2.2 聚类数目的选取

用户和项目的聚类数目对算法性能的有着不同的影响, 取用户和项目的聚类数目分别为 10、20、30、40、50、60 时, MAE 值变化如图 5 和图 6 所示.

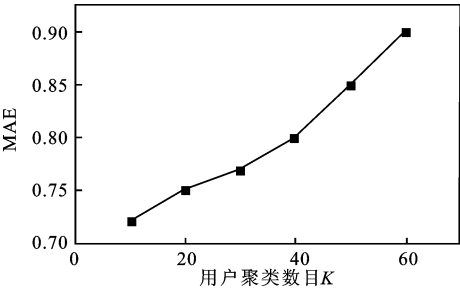


图 4 用户聚类数对 MAE 的影响

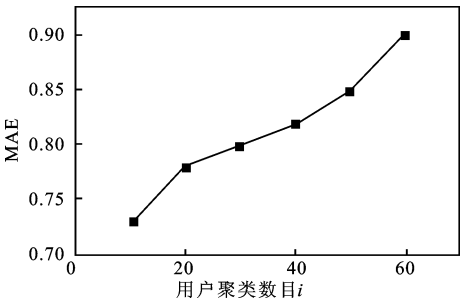


图 5 项目聚类数对 MAE 的影响

根据图 4 和图 5 可知, 随着聚类数的增加, MAE 值会随之变化, 所以正确的选择用户和项目的聚类数会直接影响到推荐的结果. 根据实验结果, 选择聚类数目和项目的值为 10 时最合适. 所以接下来的实验取用户和项目的聚类数目的值为 10.

4.2.3 实验结果

将本文算法和传统的协同过滤算法、文献[3]算法作比较, 在用户和项目聚类确定的情况下, 将邻居集的个数从 10 递增至 50, 实验结果如图 6 所示.

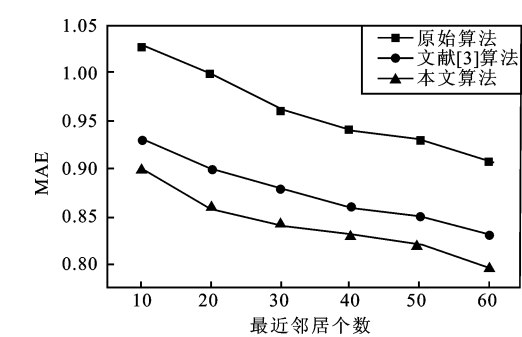


图 6 三种算法的性能比较

根据图 6 可知,随着邻居数目的增加,本文提出的算法具有最小的 MAE 值.这是由于本文首先采用联合聚类的方法对用户和项目进行联合聚类,然后对评分矩阵就行平滑填充,很大的程度上解决了数据稀疏性的问题,提高了相似性的精度;其次利用改进后的混合蛙跳算法求得最近邻居集,提高了推荐的精度,使得推荐的结果更加的精确.因此,与传统的协同过滤算法以及文献[3]的算法相比较,本文所提出的基于改进混合蛙跳联合聚类算法预测结果更加准确,提高了推荐系统的质量.

5 结束语

传统的协同过滤算法存在数据稀疏性和冷启动的问题.本文提出的基于混合蛙跳算法联合聚类的协同过滤算法,通过对用户和项目进行联合聚类后对评分矩阵进行平滑填充,有效的缓解了数据稀疏性和冷启动问题,同时利用改进的混合蛙跳算法求得最近邻居集,提高了推荐的质量.联合聚类的用户和项目的聚类的选择还是有商酌的余地,本文只是在改变一个聚类数值进行实验的,并没有在同时改变的情况下进行实验,这也将是下一步所研究的问题.

参考文献:

[1] 吴湖,王永吉,王哲,等.两阶段联合聚类协同过滤算法

[J]. 软件学报,2010,21(5):1042-1054.

[2] 查文琴. 基于用户聚类的项目多内容协同过滤推荐[D]. 合肥:合肥工业大学,2009.

[3] 邵琳琳. 基于混合蛙跳模糊聚类的电子商务协同过滤推荐[J]. 科学技术与工程,2013,13(12):3452-3456.

[4] 韦素云,肖静静,业宁. 基于联合聚类平滑的协同过滤算法[J]. 计算机研究与发展,2013, 50 (Sup2): 163-169.

[5] 李海霞. 基于蚁群聚类的电子商务个性化推荐算法研究[D]. 山东:山东师范大学,2014.

[6] 王联国,韩晓慧,宋磊. 基于改进混合蛙跳-K 均值聚类算法的无功电压控制分区[J]. 传感器与微系统,2013, 32(6):18-21.

[7] 王明文,陶红亮,熊小勇. 双向聚类迭代的协同过滤推荐算法[J]. 中文信息学报,2008,22(4):61-65.

[8] 黄少滨,杨欣欣,申林山,等. 高阶异构数据模糊联合聚类算法[J]. 通信学报,2014, 35(6):15-24.

[9] 王鹏,杨士强,刘志强. 信息论联合聚类算法及其在视频镜头聚类中的应用[J]. 计算机学报,2005,28(10): 1692-1699.

[10] 徐庆征,王磊,何宝民,等. 基于当前最优解的反向差分进化算法求解函数优化问题[J]. 应用科学学报,2011, 29(3):308-315.

[11] 贺毅朝,王熙照,刘坤起,等. 差分演化的收敛性分析与算法改进[J]. 软件学报,2010,21(5):875-885.

[12] 张强,许少华,李盼池. 基于量子蛙跳算法和过程神经网络的抽油机故障诊断[J]. 中国机械工程,2014,25 (12):1609-1615.

作者简介:

喻金平 男,(1964-),教授.研究方向为数据挖掘.

张 勇(通讯作者) 男,(1991-),硕士.研究方向为数据挖掘、群体智能. E-mail:penny_1399@163. com.

廖列法 男,(1975-),博士,教授.研究方向为组织学习、电子商务、推荐系统.

梅宏标 男,(1976-),博士,副教授.研究方向为大规模仿真系统工程.