协同过滤推荐系统中的用户博弈

徐 蕾[1)] 杨 成[2)] 姜春晓[1)] 任 勇[1)]

[1)](清华大学电子工程系 北京 100084)

[2)](中国传媒大学信息工程学院 北京 100024)

摘 要 在以协同过滤算法为核心的推荐系统中,一个用户能否获得高质量的推荐不仅取决于用户自身是否积极地参与项目评分,还取决于其他用户是否能提供足够多的评分.由于对项目评分是需要付出成本的,理性的用户总是希望以尽可能少的评分换取高质量的推荐.该文用博弈论的方法对协同过滤系统中的用户评分行为进行分析.考虑到一个用户通常无法观察到其他用户的评分和得到的推荐,该文将用户间的交互建模为不完全信息博弈,并引入"满足均衡"的概念来分析该博弈.该文假定每个用户都对推荐质量有一个预期,当所有用户的预期都得到满足时,博弈即达到均衡.针对所建立的博弈模型,该文设计了一种均衡学习算法,该算法允许用户以逐渐增加评分数量的方式来寻找均衡策略.理论分析和仿真结果均表明,当所有用户对推荐质量有着相似的预期时,所提算法可收敛到满足均衡.这一分析结果可以为协作式系统中激励机制的设计提供启发.

关键词 协同过滤;博弈论;满足均衡;均衡学习;收敛条件;社交网络;社会媒体

中图法分类号 TP391 **DOI号** 10.11897/SP.J.1016.2016.01176

Game Analysis of User Participation in Collaborative Filtering Systems

XU Lei[1)] YANG Cheng[2)] JIANG Chun-Xiao[1)] REN Yong[1)]

[1)](*Department of Electronic Engineering, Tsinghua University, Beijing* 100084)

[2)](*Information Engineering School, Communication University of China, Beijing* 100024)

Abstract User participation, i.e. providing ratings to the recommendation server, is of vital importance for the success of collaborative filtering-based recommendation systems. As the name collaborative suggests, whether a user can get high-quality recommendations depends not only on the user himself/herself but also on other users. However, due to the rating cost, rational users prefer to provide as few ratings as possible. In this paper, we model the interactions among users as a game with incomplete information and apply the notion of satisfaction equilibrium (SE) to the proposed game. Every user is assumed to have an expectation for the recommendation quality, and when all users' expectations are satisfied, a SE of the game is achieved. We design a behavior rule which allows users to achieve a SE via iteratively rating items. Theoretical analysis and simulation results demonstrate that, if all users have moderate expectations for recommendation quality and satisfied users are willing to provide more ratings, then all users can get satisfying recommendations without providing many ratings. We hope the game analysis presented in this paper can provide some implications for designing mechanisms to encourage user participation.

Keywords collaborative filtering; game theory; satisfaction equilibrium; equilibrium learning; convergence condition; social network; social media

---

收稿日期:2014-10-01;**在线出版日期**:2015-07-23.**本课题得到国家自然科学基金**(61271267,61471025)、**高等学校博士学科点专项科研基金**(20110002110060)**资助**.**徐 蕾**,女,1986年生,博士,主要研究方向为网络信息共享、博弈论应用.E-mail:xu_l04@163.com.**杨 成**,男,1974年生,博士,教授,主要研究领域为信息安全.**姜春晓**,男,1987年生,博士,主要研究方向为网络信息共享、博弈论应用.**任 勇**,男,1963年生,博士,教授,主要研究领域为信息共享、复杂系统.

1 引　言

作为一种有效的信息过滤手段，个性化推荐已广泛应用于电子商务、视频分享、在线社交等领域. 协同过滤(collaborative filtering)是目前推荐系统最常用的一类算法[1]，其基本思想是分析大量用户对项目的“评分”以发现用户偏好的相似性，然后向用户推荐符合其偏好的项目. 这种推荐算法能否取得良好效果很大程度上取决于用户是否积极地参与项目评分. 用户对项目的评分主要基于用户自身的体验. 由于给项目评分是需要付出“成本”的——评分过程会占用一些时间、评分数据可能泄露用户的隐私，用户通常不会对其体验过的所有项目都进行评分. 若用户出于成本考虑只提供很少的评分数据，则推荐服务器就无法准确计算用户之间的相似性，进而无法为用户提供高质量的推荐.

为了鼓励用户参与评分，推荐服务器可向用户提供奖金、积分等形式的激励. 目前在协同过滤的研究中对激励机制的讨论还比较少见，但在 P2P 资源共享[2-3]、参与感知[4-6]、众包[7]等类似的问题场景中已有较多关于激励机制的研究. 在个性化推荐系统中，除了奖金等形式的外部激励，推荐结果本身即可视为一种促使用户对项目评分的内部激励——用户若要获得准确的推荐，则有必要向推荐服务器提供足够的评分数据以清楚地表达自己的偏好. 在没有外部激励的情况下，用户可否自发地提供足够多的评分数据以使推荐服务器产生令人满意的推荐？如果可以，用户在选择项目进行评分时应如何决策？本文拟通过研究用户之间的“博弈”回答上述问题.

如前所述，用户对项目评分是有成本的. 用户提供的评分越多，付出的成本就越高. 用户在决定是否对一个项目进行评分时，需仔细权衡评分成本和推荐质量. 此外，推荐服务器采用“协同”过滤算法计算推荐，这意味着一个用户获得的推荐不仅与用户自己提供的评分有关，还与其他用户提供的评分有关. 换言之，不同用户的评分行为是相互影响的. 若假定用户是理性的，即每个用户都希望以尽可能低的评分成本获得满意的推荐，那么可以认为推荐系统中的各用户是以推荐服务器为媒介进行着某种形式的博弈，因此本文考虑用博弈论[8]的方法对用户的评分行为进行分析.

目前在协同过滤的相关研究中，博弈论的应用并不多见. Halkidi 等人在文献[9]中用博弈论的方法分析了协同过滤推荐系统中的隐私保护问题. 在他们研究的博弈模型中，用户向推荐服务器提供的评分向量被视为用户的策略. 为避免评分数据泄露隐私，用户可向推荐服务器提供虚假评分，但这会降低推荐准确率. 用户面临的问题是如何修改评分才能在不明显降低推荐准确率的前提下最大限度地保留隐私. 为寻找博弈的均衡[8]，Halkidi 等人将用户与推荐服务器之间的交互建模为一个迭代过程. 在每一轮迭代中，每个用户可利用其他用户在前一轮迭代提供的评分来确定自己当前的最优策略. 但在实际的推荐系统中，用户通常不会反复更新自己的评分，并且用户也无法全面准确地获得其他用户的评分信息，因而上述博弈分析的实际意义有待商榷.

考虑到在实际的推荐系统中，一个用户既无法完整的观察到其他用户的评分，也无法了解推荐服务器为其他用户提供了怎样的推荐，因此，与 Halkidi 等人的研究不同，本文将用户间的交互建模为一种不完全信息博弈，并应用“满足均衡”(satisfaction equilibrium)的概念来分析该博弈的均衡. 满足均衡描述的是这样一种状态：博弈中所有参与者的个体约束条件都得到满足. Ross 和 Chaib-draa[10]最先提出满足均衡这一概念，用于解决不完全信息博弈的纳什均衡(Nash equilibrium)难以分析的问题. 为将满足均衡应用于协同过滤系统，本文假定每个用户都对推荐结果的质量有一个预期，当实际的推荐质量高于预期时，用户即感到“满足”. 不同用户的预期一般是不同的. 当所有用户的预期都得到满足时，用户间的博弈达到满足均衡.

基于所建立的博弈模型，本文提出了一种均衡学习算法. 所提算法参考了 perlaza 等人[11-14]在研究满足均衡学习算法时提出的基于迭代的行动规则，但现有研究主要是针对无线通信网络中 QoS(Quality of Service)配置问题的，与本文研究的协同过滤问题不同，因而现有的学习算法不能直接用于本文构建的博弈模型. 结合推荐服务器的工作原理，本文定义了如下的行动规则：在每一轮迭代中，用户根据当前所得推荐是否达到预期以决定是否选择新的项目进行评分. 换言之，本文设计的均衡学习算法是让用户以逐渐增加评分数量的方式来寻找均衡策略. 在适当的简化假设下，本文对所提算法的收敛性进行了理论分析，并通过一系列仿真实验对理论分析的结果进行了验证.

本文第 2 节介绍本文所提的博弈模型并给出满足均衡的描述；第 3 节介绍均衡学习算法并给出算

法收敛性的理论分析；第 4 节介绍仿真实验的设计并对仿真结果进行分析；第 5 节给出结论.

2 博弈模型

2.1 场景描述

在一个推荐系统中，用户集合为$\mathcal{N}=\{1,2,\cdots,N\}$，项目集合为 $S=\{s_1,s_2,\cdots,s_M\}$. 用 S_i表示用户 i 体验过的项目的集合，用 $\widetilde{S}_i$表示用户 i 给出了评分的项目集合，则有 $\widetilde{S}_i\subseteq S_i\subseteq S$. 用户 i 对 $\widetilde{S}_i$中的项目评分，相当于向推荐服务器提供了评分向量 $\boldsymbol{r}_i=(r_{i1},r_{i2},\cdots,r_{iM})$. 本文规定：对任意 $j\in\{1,2,\cdots,M\}$，若$s_j\in\widetilde{S}_i$，则 $0<r_{ij}\leq r_{\max}$；若$s_j\notin\widetilde{S}_i$，则有$r_{ij}=0$. 所有用户提供的评分构成一个“用户-项目”评分矩阵 $\boldsymbol{R}=[r_{ij}]_{N\times M}$.

在得到评分矩阵后，推荐服务器应用某种协同过滤算法预测矩阵中的未知评分($r_{ij}=0$)，而后根据预测结果向用户推荐项目. 用 $\hat{\boldsymbol{r}}_i=(\hat{r}_{i1},\hat{r}_{i2},\cdots,\hat{r}_{iM})$表示与用户 i 对应的预测结果，其中 $\hat{r}_{ij}$ ($j=1,2,\cdots,M$)定义如下：

$$\hat{r}_{ij}=\begin{cases}r_{ij}, & \text{如果 } r_{ij}\neq 0;\\ f_{ij}^{CF}(\boldsymbol{R}), & \text{如果 } r_{ij}=0\end{cases}\tag{1}$$

$f_{ij}^{CF}(\boldsymbol{R})$表示预测的评分由全体用户的评分和所采用的协同过滤算法决定. 例如，若采用基于用户的最近邻协同过滤算法，$f_{ij}^{CF}(\boldsymbol{R})$可按如下方式计算：

$$f_{ij}^{CF}(\boldsymbol{R})=\frac{\sum\limits_{k\in Neighbour(i)}F_{sim}(i,k)\cdot r_{kj}}{\sum\limits_{k\in Neighbour(i)}|F_{sim}(i,k)|}\tag{2}$$

式中：$Neighbour(i)$表示与用户 i 最相似的若干个用户；$F_{sim}(i,k)$表示用户 i 与用户 k 的相似度，可用 Pearson 相关系数或余弦相似度度量[1]. 式(2)的含义是：找出与用户 i 具有相似偏好的若干用户，然后根据这些用户对项目 s_j的偏好程度来预测用户 i 对项目 s_j的偏好程度. 在实际的推荐系统中，推荐服务器会根据 $\hat{\boldsymbol{r}}_i$生成用户 i 的推荐结果，即推荐服务器将预测评分最高的若干个项目推荐给用户. 为便于分析，本文假定推荐服务器直接将 $\hat{\boldsymbol{r}}_i$返回给用户.

用户 i 在得到 $\hat{\boldsymbol{r}}_i$后，会根据自身的兴趣对这一结果进行评估. 用 $\boldsymbol{p}_i=(p_{i1},p_{i2},\cdots,p_{iM})$表示用户 i 的兴趣，其中 p_{ij} ($j\in\{1,2,\cdots,M\}$)表示用户 i 对项目 s_j的偏好度. 规定：$0\leq p_{ij}\leq r_{\max}$，并且若 $s_j\in\widetilde{S}_i$，则 $p_{ij}=r_{ij}$，即用户给出的项目评分等于用户对该项目的偏好度. 若 $s_j\notin\widetilde{S}_i$，可将 p_{ij}理解为用户对项目的“潜在”评分，即如果用户在将来的某一时刻对 s_j评分，他给出的评分就会是 p_{ij}. 为每个用户定义函数 $g_i:\mathrm{R}^M\to[0,1]$来度量推荐结果 $\hat{\boldsymbol{r}}_i$的质量：

$$g_i(\hat{\boldsymbol{r}}_i)=1-\frac{\sqrt{\sum\limits_{j=1}^{M}(\hat{r}_{ij}-p_{ij})^2}}{r_{\max}\sqrt{M}}\tag{3}$$

式中，右侧第 2 项中的$\sqrt{\sum\limits_{j=1}^{M}(\hat{r}_{ij}-p_{ij})^2}$表示预测评分向量 $\hat{\boldsymbol{r}}_i$与用户兴趣向量 $\boldsymbol{p}_i$的距离，$r_{\max}\sqrt{M}$表示这一距离可能取得的最大值. $g_i(\hat{\boldsymbol{r}}_i)$的值越大表示推荐结果与用户兴趣的匹配程度越高，即推荐质量越高.

从式(1)～(3)可以看出，一个用户所得推荐的质量与其他用户提供的评分密切相关. 协同过滤系统中的每个用户实际上是以向推荐服务器提供评分的方式与其他用户进行着博弈. 下一小节给出该博弈的满足式表述(satisfaction form)[11].

2.2 “满足”博弈

2.2.1 参与者和行动

本文将集合$\mathcal{N}$中的所有用户视为博弈的参与者，将 $\widetilde{S}_i$视为用户 i 的行动(action)，即 $a_i=\widetilde{S}_i$. 用$\mathcal{A}_i$表示用户 i 的行动空间(action space). 所有用户共享同一个行动空间，即对任意 $i\in\mathcal{N}$，都有$\mathcal{A}_i=\{A^{(1)},A^{(2)},\cdots,A^{(K)}\}$，其中 $A^{(k)}\subseteq S$ 且 $A^{(k)}\neq\varnothing$ ($k=1,2,\cdots,K$)，$K=2^{|S|}-1$. 用户 i 从$\mathcal{A}_i$中选择行动时遵循特定的概率分布 $\boldsymbol{\pi}_i=(\pi_i^{(1)},\pi_i^{(2)},\cdots,\pi_i^{(K)})$，其中 $\pi_i^{(k)}$表示用户 i 选择行动 $A^{(k)}$ 的概率. 不同用户的 $\boldsymbol{\pi}_i$一般是不同的.

由式(1)可知，若给定协同过滤算法，则推荐服务器返回给每个用户的推荐结果 $\hat{\boldsymbol{r}}_i$完全由评分矩阵 $\boldsymbol{R}$确定，而 $\boldsymbol{R}$是由所有用户的行动组合 $\boldsymbol{a}=(a_1,a_2,\cdots,a_N)$确定的. 为体现用户行动对推荐质量的影响，可将 $g_i(\hat{\boldsymbol{r}}_i)$改写为 $g_i(\hat{\boldsymbol{r}}_i)=h_i(a_i,\boldsymbol{a}_{-i})$，其中 $\boldsymbol{a}_{-i}=(a_1,\cdots,a_{i-1},a_{i+1},\cdots,a_N)$，$h_i(\cdot)$表示从$\mathcal{A}\triangleq\mathcal{A}_1\times\mathcal{A}_2\times\cdots\times\mathcal{A}_N$到$[0,1]$的映射. 直观上理解，无论是用户 i 自己提供更多的评分还是其他用户提供更多的评分，用户 i 都能获得更好的推荐. 为度量用户提供评分的数量，定义用户 i 的评分完整度如下：

$$\sigma_i=\frac{|a_i|}{|S_i|}\tag{4}$$

由 $a_i\subseteq S_i$且 $a_i\neq\varnothing$可知 $0<\sigma_i\leq 1$. 用 σ_{-i}表示其他用户的评分完整度的平均值，即

$$\sigma_{-i}=\frac{1}{N-1}\sum_{j\in\mathcal{N},j\neq i}\sigma_j\tag{5}$$

引入 σ_i和σ_{-i}之后，可将 $h_i(a_i,\boldsymbol{a}_{-i})$改写为 $h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)$. 函数 $h(\cdot;\boldsymbol{p}_i)$以 $\boldsymbol{p}_i$作为参数，以 σ_i和 σ_{-i}作为输

入变量. 本文对 $h(\cdot;\boldsymbol{p}_i)$ 的单调性做出如下假设.

假设 1.　对任意用户 $i\in\mathcal{N}$,以下两个不等式对所有 $\sigma_i\in(0,1]$ 和 $\sigma_{-i}\in(0,1]$ 都成立:

(1) $\dfrac{\partial h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)}{\partial\sigma_i}>0$;

(2) $\dfrac{\partial h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)}{\partial\sigma_{-i}}>0$.

上述假设表明,当每个用户都对其体验过的所有项目给出评分时,用户可得到最好的推荐. 用 $\Gamma_i^{\max}$ 表示最优的推荐质量,则有 $\Gamma_i^{\max}=h(1,1;\boldsymbol{p}_i)$. 在引言部分提到过,用户对项目评分是需要付出成本的. 用 $c_i(a_i)$ 表示用户 i 选择行动 a_i 时所付出的成本. 对任意 $a_i'\in A_i$ 和 $a_i''\in A_i$,若 $a_i'\subset a_i''$,则有 $c_i(a_i')<c_i(a_i'')$.

2.2.2　满足式表述

由于存在评分成本,用户通常不会对其体验过的所有项目都进行评分,这就意味着最优推荐 $\Gamma_i^{\max}$ 很难实现. 假定每个用户 i 对推荐质量有一个低于最优值的预期 Γ_i. 给定用户的行动组合 $\boldsymbol{a}$,只要 $h_i(\boldsymbol{a})\geq\Gamma_i$,用户 i 即可满足. 定义映射 $f_i:\mathcal{A}_{-i}\to 2^{\mathcal{A}_i}$ ($\mathcal{A}_{-i}\triangleq\mathcal{A}_1\times\cdots\times\mathcal{A}_{i-1}\times\mathcal{A}_{i+1}\times\cdots\times\mathcal{A}_N$) 如下:

$$f_i(\boldsymbol{a}_{-i})=\{a_i\in\mathcal{A}_i:h_i(a_i,\boldsymbol{a}_{-i})\geq\Gamma_i\}\tag{6}$$

映射 f_i 通常被称为 correspondence[11].

基于上述讨论,本文用如下的三元组描述所建立的博弈模型:

$$\hat{G}_{CF}=(\mathcal{N},\{\mathcal{A}_i\}_{i\in\mathcal{N}},\{f_i\}_{i\in\mathcal{N}})\tag{7}$$

上述形式称为博弈的满足式表述. Perlaza 等人[11]在研究分布式自配置网络中的 QoS 保障问题时最先正式定义了这一形式的博弈.

2.2.3　满足均衡

与博弈的满足式表述对应的均衡称为满足均衡(Satisfaction Equilibrium,SE). SE 的规范定义如下[11].

定义 1.　满足均衡. 给定博弈的满足式表述 $\hat{G}_{CF}=(\mathcal{N},\{\mathcal{A}_i\}_{i\in\mathcal{N}},\{f_i\}_{i\in\mathcal{N}})$,一个行动组合 $\boldsymbol{a}^+$ 若满足 $\forall i\in\mathcal{N}$,有 $a_i^+\in f_i(\boldsymbol{a}_{-i}^+)$,则称 $\boldsymbol{a}^+$ 为该博弈的满足均衡.

因本文假定对任意 $i\in\mathcal{N}$ 都有 $\Gamma_i<\Gamma_i^{\max}$,所以 $\boldsymbol{a}^{\max}\triangleq(S_1,S_2,\cdots,S_N)$ 是 $\hat{G}_{CF}$ 的一个 SE. $\boldsymbol{a}^{\max}$ 要求每个用户付出最高的评分成本 $c_i(S_i)$,而事实上由于 $\Gamma_i<\Gamma_i^{\max}$,用户可能并不需要用如此高的成本去换取满意的推荐结果. 本文主要考虑如何找到符合如下两个条件的满足均衡 $\boldsymbol{a}^+=(a_1^+,a_2^+,\cdots,a_N^+)$:

(1) $\forall i\in\mathcal{N},a_i^+\in f_i(\boldsymbol{a}_{-i}^+)$,且 $c_i(a_i^+)\leq c_i(S_i)$;

(2) 至少存在一个用户不需要对其体验过的所有项目都给予评分,即 $\exists i\in\mathcal{N},c_i(a_i^+)<c_i(S_i)$ 成立.

3　均衡学习

在上文建立的博弈模型 $\hat{G}_{CF}$ 中,每个用户在行动空间上的概率分布 $\boldsymbol{\pi}_i$、对推荐质量的预期 Γ_i、兴趣向量 $\boldsymbol{p}_i$、评分成本 c_i 都是其私有信息(private information),因而直接分析该博弈的均衡是很困难的. 一种可行的方法是设计某种行为规则,让用户在迭代交互的过程中依据该规则不断调整自己的策略,最终学出一种均衡策略. 结合推荐服务器的工作原理,本文设计了一种学习算法,该算法允许用户以逐渐增加评分数量的方式寻找均衡策略. 本节首先介绍算法的基本流程,然后对算法的收敛性进行分析.

3.1　均衡学习算法

在推荐系统中,用户与推荐服务器的交互是长期的——用户不断向推荐服务器提供评分数据,推荐服务器则不断调整反馈给用户的推荐. 本文规定,每个用户在与推荐服务器迭代交互的过程中按如下规则行动:

初始时刻($n=0$),用户 i 计算其在行动空间上的概率分布 $\boldsymbol{\pi}_i(0)=(\pi_i^{(1)}(0),\pi_i^{(2)}(0),\cdots,\pi_i^{(K)}(0))$,然后选择行动 $a_i(0)$. 用户选择行动 $A^{(k)}$ 的概率 $\pi_i^{(k)}(0)$ 为

$$\pi_i^{(k)}(0)=\begin{cases}\beta_i(0)/\alpha^{c_i(A^{(k)})}, & \text{如果 } A^{(k)}\subseteq S_i\\ 0, & \text{其他}\end{cases}\tag{8}$$

其中,参数 $\alpha(\alpha>1)$ 表示用户对评分成本的重视程度. α 越小,用户越有可能选择较多的项目进行评分. 归一化因子 $\beta_i(0)$ 按如下方式计算:

$$\beta_i(0)=\frac{1}{\sum\limits_{k:A^{(k)}\subseteq S_i}\alpha^{-c_i(A^{(k)})}}\tag{9}$$

当所有用户选择完初始行动后,推荐服务器根据得到的评分矩阵 $\boldsymbol{R}(0)$ 计算推荐,然后将 $\hat{\boldsymbol{r}}_i(0)$ 返回给用户 i. 在此之后,用户以迭代的方式与推荐服务器交互.

在第 n 次迭代开始时($n=1,2,\cdots$),用户 i 首先判断目前的推荐 $\hat{\boldsymbol{r}}_i(n-1)$ 是否已达预期. 定义指示变量 $v_i(n-1)$ 如下:

$$v_i(n-1)=\begin{cases}1, & \text{如果 } g_i(\hat{\boldsymbol{r}}_i(n-1))\geq\Gamma_i\\ 0, & \text{其他}\end{cases}\tag{10}$$

用户根据 $v_i(n-1)$ 的取值更新概率分布 $\boldsymbol{\pi}_i(n)=(\pi_i^{(1)}(n),\pi_i^{(2)}(n),\cdots,\pi_i^{(K)}(n))$,然后选择行动 $a_i(n)$. 需要特别指出的是,推荐服务器会利用用户以往提

供的全部评分来计算推荐，所以从推荐结果来看，用户在第 n 次迭代中选择若干个新项目（记为 $\widetilde{S}_i^{\text{new}}(n)$）进行评分，其效果等价于用户在初始时刻对集合 $\widetilde{S}_i^{\text{new}}(n)\bigcup\cdots\bigcup\widetilde{S}_i^{\text{new}}(1)\bigcup\widetilde{S}_i(0)$ 中的项目进行了评分. 因此，本文将 $a_i(n)$ 定义为从初始时刻至第 n 次迭代结束时，用户 i 已经评过分的所有项目的集合. 易知 $a_i(n)\supseteq a_i(n-1)$.

如果当前的推荐质量未达预期，即 $v_i(n-1)=0$，用户 i 可能将这一结果归结为以下两种原因之一：自己提供的评分太少；自己已提供足够多的评分，但其他用户提供的评分太少. 若是前者，用户会对更多的项目进行评分；若是后者，用户倾向于不选择新的项目评分. 行动选择概率 $\pi_i^{(k)}(n)\triangleq\Pr(a_i(n)=A^{(k)})$ 按如下方式计算：

$$\pi_i^{(k)}(n)=\begin{cases}\sigma_i(n-1), & \text{如果 } A^{(k)}=a_i(n-1)\\ \beta_i(n)/\alpha^{c_i(A^{(k)})}, & \text{如果 } a_i(n-1)\subset A^{(k)}\subseteq S_i\\ 0, & \text{其他}\end{cases}\tag{11}$$

其中，$\sigma_i(n-1)$ 表示用户 i 当前的评分完整度：

$$\sigma_i(n-1)=\frac{|a_i(n-1)|}{|S_i|}\tag{12}$$

$\sigma_i(n-1)$ 越高，表明用户 i 已提供的评分越多，则用户继续提供评分的概率越低. 归一化因子 $\beta_i(n)$ 定义如下：

$$\beta_i(n)=\frac{1-\sigma_i(n-1)}{\sum\limits_{k;a_i(n-1)\subset A^{(k)}\subseteq S_i}\alpha^{-c_i(A^{(k)})}}\tag{13}$$

如果当前的推荐质量已达预期，即 $v_i(n-1)=1$，则用户很有可能不再提供更多的评分. 此时，概率 $\pi_i^{(k)}(n)$ 按如下方式确定：

$$\pi_i^{(k)}(n)=\begin{cases}\mu, & \text{如果 } A^{(k)}=a_i(n-1)\\ \beta_i(n)/\alpha^{c_i(A^{(k)})}, & \text{如果 } a_i(n-1)\subset A^{(k)}\subseteq S_i\\ 0, & \text{其他}\end{cases}\tag{14}$$

其中，参数 μ 表示用户维持原有行动的概率，通常有 $0.5<\mu\leq 1$. 归一化因子 $\beta_i(n)$ 定义为

$$\beta_i(n)=\frac{1-\mu}{\sum\limits_{k;a_i(n-1)\subset A^{(k)}\subseteq S_i}\alpha^{-c_i(A^{(k)})}}\tag{15}$$

当所有用户选择了行动后，推荐服务器根据得到的评分矩阵 $\boldsymbol{R}(n)$ 计算推荐，然后将 $\hat{\boldsymbol{r}}_i(n)$ 返回给用户 i，之后进入下一轮迭代. 若经过 $n_s(n_s>0)$ 次迭代之后，所有用户都得到了满意的推荐，则迭代终止，称学习过程收敛于满足均衡 $\boldsymbol{a}^+=(a_1(n_s),a_2(n_s),\cdots,a_N(n_s))$. 我们将上述行为规则归纳为算法 1.

算法 1. 满足均衡学习算法.

1. $n=0$：$\forall k\in\{1,2,\cdots,K\}$，

$$\pi_i^{(k)}(0)=\begin{cases}\beta_i(0)/\alpha^{c_i(A^{(k)})}, & \text{如果 } A^{(k)}\subseteq S_i\\ 0, & \text{其他}\end{cases},$$

其中，$\beta_i(0)=\dfrac{1}{\sum\limits_{k;A^{(k)}\subseteq S_i}\alpha^{-c_i(A^{(k)})}}$.

2. 选择 $a_i(0)\sim\boldsymbol{\pi}_i(0)$；
3. FOR ALL $n>0$ DO：
4. 更新 $\boldsymbol{\pi}_i(n)$：$\forall k\in\{1,2,\cdots,K\}$，

$$\pi_i^{(k)}(n)=\begin{cases}\gamma_i(n), & \text{如果 } A^{(k)}=a_i(n-1)\\ \beta_i(n)/\alpha^{c_i(A^{(k)})}, & \text{如果 } a_i(n-1)\subset A^{(k)}\subseteq S_i\\ 0, & \text{其他}\end{cases},$$

其中，$\gamma_i(n)=\begin{cases}\sigma_i(n-1), & \text{如果 } v_i(n-1)=0\\ \mu, & \text{如果 } v_i(n-1)=1\end{cases}$，

$$\beta_i(n)=\frac{1-\gamma_i(n)}{\sum\limits_{k;a_i(n-1)\subset A^{(k)}\subseteq S_i}\alpha^{-c_i(A^{(k)})}}.$$

5. 选择 $a_i(n)\sim\boldsymbol{\pi}_i(n)$
6. END FOR

3.2 收敛性分析

为了分析上文所提均衡学习算法的收敛性，本小节首先给出“用户状态”的定义，然后推导算法的收敛条件.

3.2.1 用户状态

如图 1 所示，给定用户预期 Γ_i，等式 $h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)=\Gamma_i$ 定义了 $\sigma_i\sim\sigma_{-i}$ 平面上的一条曲线段. 由假设 1 可知，只有当 σ_i 和 σ_{-i} 分别高于某个阈值时，用户 i 才有可能得到满足. 阈值 $\sigma_{i,\min}$ 和 $\sigma_{-i,\min}$ 分别由 $h(\sigma_{i,\min},1;\boldsymbol{p}_i)=\Gamma_i$ 和 $h(1,\sigma_{-i,\min};\boldsymbol{p}_i)=\Gamma_i$ 确定. $\sigma_{i,\min}$ 和 $\sigma_{-i,\min}$

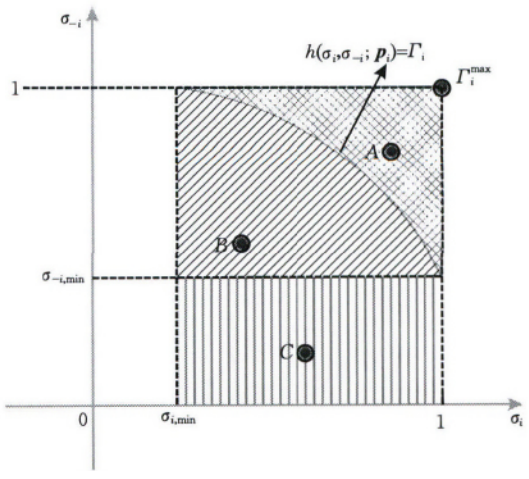


图 1 用户状态示意图（网格区域：“满足”；斜线区域：“接近满足”；竖线区域：“远离满足”）

分别代表了对用户 i 自身提供评分数量和对其他用户提供评分数量的最低要求.

在学习过程中，每个用户的评分完整度随着迭代次数的增加而增加，即有 $\sigma_i(n)\geq\sigma_i(n-1)$. 根据式(12)，定义 $\sigma_{-i}(n-1)$ 如下：

$$\sigma_{-i}(n-1)=\frac{1}{N-1}\sum_{j\in N,j\neq i}\frac{|a_j(n-1)|}{|S_j|} \tag{16}$$

易知 $\sigma_{-i}(n)\geq\sigma_{-i}(n-1)$. 假定存在某个 $n_0(n_0\geq 1)$ 使得 $\sigma_i(n_0)\geq\sigma_{i,\min}$ 对所有 i 均成立，则从第 n_0+1 次迭代开始，每个用户 i 处于以下 3 种状态之一：

(1) 满足(Satisfied). 用户 i 已得到满意的推荐，即 $h(\sigma_i(n-1),\sigma_{-i}(n-1);\boldsymbol{p}_i)\geq\Gamma_i$. 用户进入满足状态后会一直保持该状态，因为随着迭代次数的增加，σ_i 和 σ_{-i} 会增加或保持不变，且由假设 1 可知 $h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)$ 亦增加或保持不变.

(2) 接近满足(Proximity to satisfied). 用户 i 尚未得到满意的推荐，即 $h(\sigma_i(n-1),\sigma_{-i}(n-1);\boldsymbol{p}_i)<\Gamma_i$，且用户 i 尚未对 S_i 中的所有项目都进行评分，即 $\sigma_i(n-1)<1$，但此时其他用户提供的评分数量已达到用户 i 的最低要求，即 $\sigma_{-i}(n-1)\geq\sigma_{-i,\min}$. 在这种情况下，即使其他用户不再提供更多的评分，用户 i 也可以通过增加自己的 σ_i 进入满足状态.

(3) 远离满足(Far from satisfied). 用户 i 尚未得到满意的推荐，并且其他用户提供的评分数量未达到用户 i 的最低要求，即 $\sigma_{-i}(n-1)<\sigma_{-i,\min}$. 在这种情况下，如果其他用户在后续的迭代过程中能提供足够多的评分，那么用户 i 可进入接近满足状态，否则用户将永远停留在远离满足状态.

分别用 Z_S,Z_P,Z_F 表示上述 3 种状态. 用户 i 在第 $n(n\geq n_0)$ 次迭代开始时的状态记为 $z_i(n)$.

3.2.2　收敛条件

在第 $n(n\geq n_0)$ 次迭代开始时，可根据用户状态将全体用户分为两组：已满足的用户 $\mathcal{N}_S(n)\triangleq\{i\mid i\in\mathcal{N},z_i(n)=Z_S\}$，未满足的用户 $\mathcal{N}_{US}(n)\triangleq\{i\mid i\in\mathcal{N},z_i(n)=Z_P\vee z_i(n)=Z_F\}$. 随着迭代次数的增加，未满足的用户数逐渐下降. 若算法在第 n_S 次迭代开始时达到满足均衡，则有 $|\mathcal{N}_{US}(n_S)|=0$. 从上文对用户状态的定义可知，为了判断能否达到所有用户都处于满足状态的均衡，关键在于分析清楚用户评分完整度在迭代过程中的变化. 由算法 1 可知，评分完整度在一次迭代中的增量 $\Delta\sigma_i(n)\triangleq\sigma_i(n)-\sigma_i(n-1)$ 是随机的，因此很难定量分析用户状态的变化. 此处对算法 1 进行简化，然后推导简化后算法的收敛条件.

(1) 简化的均衡学习算法

初始时刻，每个用户 i 从集合 S_i 中随机选择一个项目进行评分. 因而对任一 $i\in\mathcal{N}$，都有 $\sigma_i(0)=\frac{1}{|S_i|}$. 在第 n 次迭代开始时$(n>0)$，若 $h(\sigma_i(n-1),\sigma_{-i}(n-1);\boldsymbol{p}_i)\geq\Gamma_i$，则用户 i 维持原有行动，即 $a_i(n)=a_i(n-1)$；若 $h(\sigma_i(n-1),\sigma_{-i}(n-1);\boldsymbol{p}_i)<\Gamma_i$，则用户 i 从集合 $S_i\backslash a_i(n-1)$ 中随机选择一个项目进行评分.

由上述行为规则可知：若 $i\in\mathcal{N}_S(n)$，则有 $\Delta\sigma_i(n)=0$；若 $i\in\mathcal{N}_{US}(n)$，则有 $\Delta\sigma_i(n)=\frac{1}{|S_i|}$.

(2) 两类用户

为了推导均衡学习算法收敛条件的解析表达式，除了对算法本身进行简化，本文亦做出如下假设.

假设 2.　全体用户可分为两组，分别记为 $\mathcal{N}_A$ 和 $\mathcal{N}_B$.

① $\forall i\in\mathcal{N}_A,\Gamma_i=\eta_A\Gamma_i^{\max}$，其中 $0<\eta_A<1$；

② $\forall i\in\mathcal{N}_B,\Gamma_i=\eta_B\Gamma_i^{\max}$，其中 $\eta_A<\eta_B<1$；

③ $\forall i\in\mathcal{N},|S_i|=M_0$，其中 M_0 是一个常数且有 $1\leq M_0<|S|$；

④ $\forall i\in\mathcal{N}$，给定 $\eta_i\in\{\eta_A,\eta_B\}$，$\sigma_i$ 和 σ_{-i} 满足如下等式：

$$\sigma_i^2+\sigma_{-i}^2=2\eta_i^2 \tag{17}$$

其中，$\frac{1}{M_0}\leq\sigma_i\leq 1,\frac{1}{M_0}\leq\sigma_{-i}\leq 1$.

根据上述假设，$\sigma_{i,\min}$ 和 $\sigma_{-i,\min}$ 可按如下方式计算：若 $0<\eta_i\leq\frac{1}{\sqrt{2}}$(图 2 中 Γ_i^A 对应的弧线)，则 $\sigma_{i,\min}=\sigma_{-i,\min}=\frac{1}{M_0}$；若 $\frac{1}{\sqrt{2}}<\eta_i<1$（图 2 中 Γ_i^B 对应的弧线），则 $\sigma_{i,\min}=\sigma_{-i,\min}=\sqrt{2\eta_i^2-1}$.

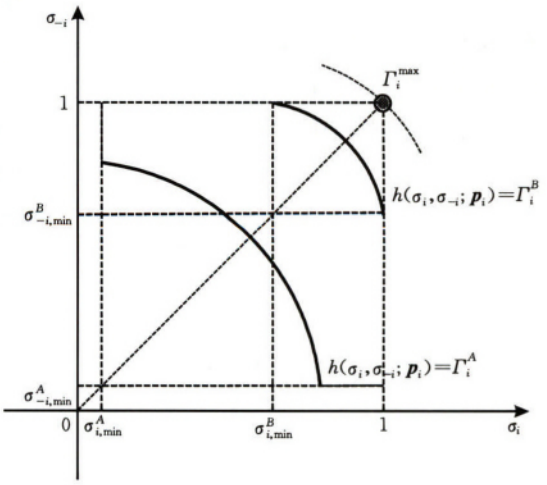


图 2　σ_i 和 σ_{-i} 之间的关系(给定 Γ_i)

(3)评分完整度的变化

基于上述简化和假设,我们可定量分析算法的收敛条件.考虑$\mathcal{N}_B$中的一个用户i.如图3所示,随着迭代的进行,用户在正方形区域$[0,1]^2$中从左下角向右上方移动.由简化的均衡学习算法可知,第1次迭代开始时,用户i位于点$\left(\frac{1}{M_0},\frac{1}{M_0}\right)$.在有用户进入满足状态之前,如下两个等式对所有用户均成立:

$$\Delta\sigma_i(n)=\frac{1}{M_0} \tag{18}$$

$$\Delta\sigma_{-i}(n)=\sigma_{-i}(n)-\sigma_{-i}(n-1)=\frac{1}{M_0} \tag{19}$$

$\mathcal{N}_A$中的用户对推荐结果有相对较低的预期,因此这些用户先于$\mathcal{N}_B$中的用户进入满足状态,即存在某个$n_A\in\{1,2,\cdots,M_0-1\}$使得$\mathcal{N}_S(n_A)=\mathcal{N}_A$,$\mathcal{N}_{US}(n_A)=\mathcal{N}_B$.在第$n_A$次迭代开始前,用户始终沿着直线$\sigma_{-i}=\sigma_i$移动.

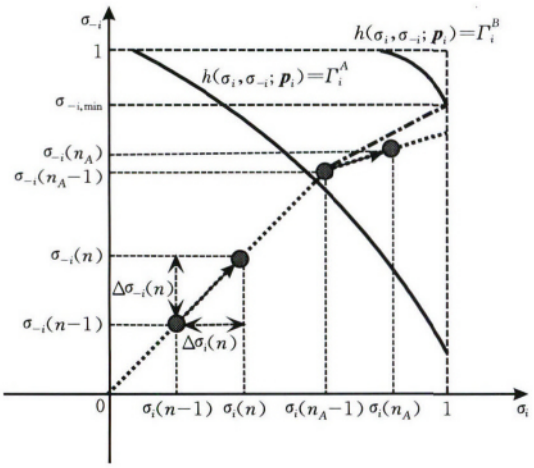


图3 用户状态变化示意

第n_A次迭代开始时,用户i位于点$\left(\frac{n_A}{M_0},\frac{n_A}{M_0}\right)$.若$z_i(n_A)=Z_P$,则用户$i$可在若干次迭代后进入$Z_S$状态.下面重点分析另一种情况,即$z_i(n_A)=Z_F$.在第$n_A$次迭代中,$\mathcal{N}_B$中的每个用户选择一个新的项目进行评分,而$\mathcal{N}_A$中的用户不再提供新的评分,用户$i$在$\sigma_i\sim\sigma_{-i}$平面上沿着斜率为$k_B(k_B<1)$的直线移动:

$$\begin{aligned}k_B&=\frac{\Delta\sigma_{-i}(n_A)}{\Delta\sigma_i(n_A)}\\&=\frac{\frac{1}{N-1}\sum\limits_{j\in\mathcal{N}_B,j\neq i}[\sigma_j(n_A)-\sigma_j(n_A-1)]}{\frac{1}{M_0}}\\&=\frac{|\mathcal{N}_B|-1}{N-1}\end{aligned} \tag{20}$$

之后,用户i继续沿着该方向移动,直到下述两种情况之一发生:用户i进入满足状态;用户i尚未满足但已对集合S_i中的所有项目进行了评分,即$\sigma_i=1$.当第2种情况出现时,用户i将永远不能满足.其原因是$\mathcal{N}_B$中的用户对推荐结果有相同的预期,这意味着此时除用户i外$\mathcal{N}_B$中的其他用户也已提供了完整的评分,σ_{-i}无法继续增加.如图2所示,当k_B小于某个阈值$k_{\min}$时,上述第2种情况就会出现.$k_{\min}$定义如下:

$$k_{\min}=\frac{\sigma_{-i,\min}-\sigma_{-i}(n_A-1)}{1-\sigma_i(n_A-1)}=\frac{\sigma_{-i,\min}-\frac{n_A}{M_0}}{1-\frac{n_A}{M_0}} \tag{21}$$

将式(20)和(21)代入$k_B<k_{\min}$可得

$$\sigma_{-i,\min}>\frac{|\mathcal{N}_B|-1}{N-1}+\frac{N-|\mathcal{N}_B|}{N-1}\cdot\frac{n_A}{M_0} \tag{22}$$

式(22)的右半部分实际上就是如下情况对应的σ_{-i}:$\mathcal{N}_B$中的所有用户都提供了完整的评分,而$\mathcal{N}_A$中的用户只提供了能使自己得到满足的必要数量的评分.

(4)用户预期与收敛性

式(22)说明如果一个用户对推荐结果有很高的预期,即需要其他用户付出较高的评分成本,那么该用户就可能无法得到满意的推荐.由假设2可知,当用户的预期不同时,$\sigma_{-i,\min}$的定义方式不同:

① 如果$\eta_A<\eta_B\leq\frac{1}{\sqrt{2}}$,则对所有$i\in\mathcal{N}_B$都有$\sigma_{-i,\min}=\frac{1}{M_0}$.考虑到$n_A\geq1$,由式(22)可得

$$(M_0-1)(|N_B|-1)<0 \tag{23}$$

因$M_0\geq1$且$|\mathcal{N}_B|\geq1$,上述不等式不成立.也就是说,当$\eta_A<\eta_B\leq\frac{1}{\sqrt{2}}$时,不会出现$k_B<k_{\min}$的情况,算法必然收敛.

② 如果$\frac{1}{\sqrt{2}}<\eta_B<1$,则对所有$i\in\mathcal{N}_B$都有$\sigma_{-i,\min}=\sqrt{2\eta_B^2-1}$.对于$\mathcal{N}_A$中的任一用户$i$,由$z_i(n_A)=Z_S$可知$\frac{n_A}{M_0}\geq\eta_A$.进一步由式(22)可知,若式(24)成立,则算法不能收敛:

$$\sqrt{2\eta_B^2-1}>\frac{|\mathcal{N}_B|-1}{N-1}+\frac{N-|\mathcal{N}_B|}{N-1}\eta_A \tag{24}$$

基于上述讨论,本文给出如下定理.

定理1. 在假设2成立的前提下,若下述两个条件之一成立,则简化后的均衡学习算法可以收敛到博弈$\hat{G}_{CF}=(\mathcal{N},\{A_i\}_{i\in\mathcal{N}},\{f_i\}_{i\in\mathcal{N}})$的一个SE:

① $\eta_A<\eta_B\leq\frac{1}{\sqrt{2}}$；

② $\frac{1}{\sqrt{2}}<\eta_B<1$，并且

$$\sqrt{2\eta_B^2-1}\leq\frac{|\mathcal{N}_B|-1}{N-1}+\frac{N-|\mathcal{N}_B|}{N-1}\eta_A \quad (25)$$

(5) 收敛阈值

为了更好的理解用户预期对算法收敛性的影响，本文给出如下分析：

① 给定 $\rho_N\triangleq\frac{|\mathcal{N}_B|}{N}$，由式(25)可知，若下述不等式成立，则简化后的均衡学习算法无法收敛：

$$\eta_B>\sqrt{\frac{1}{2}\left[\frac{\rho_N N-1}{N-1}+\frac{(1-\rho_N)N}{N-1}\eta_A\right]^2+\frac{1}{2}} \quad (26)$$

用 θ_B 表示上述不等式的右半部分. 如图 4(a)所示，在给定 ρ_N 的条件下，θ_B 随着 η_A 的增加而增加，但 θ_B 的增长速率相对较低. 这表明当用户对推荐结果的整体预期变高时(η_A 增长)，即使两组用户在预期上的差异不显著，仍会有部分用户无法得到满足. 从图 4(a)还可以看出，给定 η_A，θ_B 随着 ρ_N 的增加而增加. 这意味着当有越来越多的用户对推荐结果抱有较高预期时，用户可期望得到更高质量的推荐.

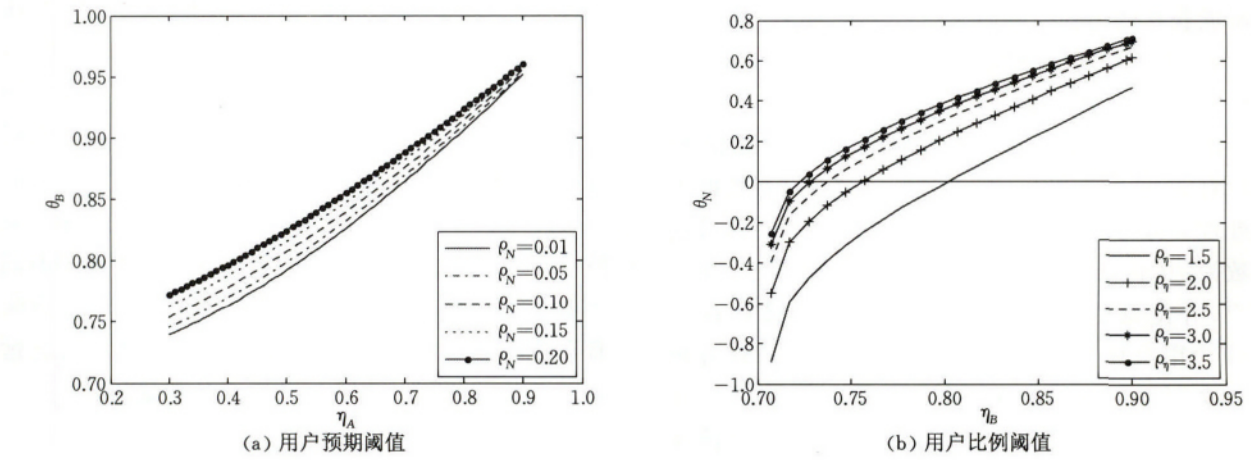
② 给定 $\rho_\eta\triangleq\frac{\eta_B}{\eta_A}$，由式(25)可知，若 $\frac{1}{\sqrt{2}}<\eta_B<1$，且下述不等式成立，则简化的均衡学习算法无法收敛：

$$\frac{|\mathcal{N}_B|}{N}<\frac{(N-1)\sqrt{2\eta_B^2-1}+1-N\frac{\eta_B}{\rho_\eta}}{N\left(1-\frac{\eta_B}{\rho_\eta}\right)} \quad (27)$$

用 θ_N 表示上述不等式的右半部分. 该不等式暗含着 $\theta_N>0$ 这一条件. 由 $\theta_N>0$ 可得

$$\eta_B>\frac{-\frac{2N}{\rho_\eta}+\sqrt{\left(\frac{2N}{\rho_\eta}\right)^2+4\left[2(N-1)^2-\frac{N^2}{\rho_\eta^2}\right](N^2-2N+2)}}{2\left[2(N-1)^2-\frac{N^2}{\rho_\eta^2}\right]} \quad (28)$$

用 $\eta_{B,\min}$ 表示式(28)的右半部分. 如图 4(b)所示，给定 ρ_η，θ_N 随着 η_B 的增加而增加. 这表明当 $\mathcal{N}_B$ 中的用户的预期变得更高时，只有当 $\mathcal{N}_B$ 中用户的数量也有所增加时，才有可能实现满足均衡. 另外，给定 η_B，θ_N 随着 ρ_η 的增加而增加，这表明当两组用户在预期上的差异变大时，可以有更多的用户预期得到高质量的推荐. 上述结论与从图 4(a)中得到的结论是一致的.



(a) 用户预期阈值　　(b) 用户比例阈值

图 4　均衡学习算法的收敛条件((a)给定 ρ_N 和 η_A，若 $\eta_B>\theta_B$，则简化后的学习算法无法收敛. 绘制曲线时，设 $0.3\leq\eta_A\leq0.9$，$N=10000$；(b)给定 ρ_η，若 $\eta_B>\eta_{B,\min}$ 且 $\rho_N<\theta_N$，则简化后的学习算法无法收敛. 绘制曲线时，设 $\frac{1}{\sqrt{2}}<\eta_B\leq0.9$，$N=10000$)

4　仿真分析

为了检验所提均衡学习算法的可行性，本文用真实的评分数据进行了一系列仿真实验. 本节首先对实验所用的数据集以及仿真参数的设置进行说明，然后对不同参数设置下的仿真结果进行分析，最后对 3.2 节给出的算法收敛条件进行验证.

4.1　数据集和参数设置

本文分别用来自 Jester 数据集[15] 和 MovieLens 数据集①的评分数据进行了仿真.

4.1.1　Jester

本文使用的 Jester 数据集包含了 24983 个用户

① http://files.grouplens.org/datasets/movielens/ml-1m.zip

对 100 个笑话的评分. 用户对笑话的评分在区间[−10,10]上取值. 若用户未给出某个笑话的评分,则用"99"表示这一未知的评分. 在所有用户中,有 7200 个用户对全部的 100 个笑话都给出了评分. 考虑到在均衡学习过程中需根据用户对所有项目的"潜在"评分来评估推荐质量(参见式(3)),本文仅选择这 7200 个用户的评分作为实验用数据. 我们将原始评分数据转换为评分矩阵 $\boldsymbol{R}=[r_{ij}]_{7200\times100}$,并将 r_{ij} 调整至区间[10.0,30.0](以 $r_{ij}=0$ 表示未知的评分).

均衡学习算法涉及的参数按如下方式设置:矩阵 $\boldsymbol{R}$ 的每一行被视为对应用户的兴趣向量 $\boldsymbol{p}_i$;假定每个用户体验过的项目总数均为 70,即 $|S_i|=70$. 为确定每个用户的 S_i,从矩阵 $\boldsymbol{R}$ 的每一行中随机选择 30 个元素置为零. 处理后的评分矩阵记为 $\boldsymbol{R}'$;按式(2)预测未知评分,设 $|Neighbour(i)|=\dfrac{N}{200}=36$. 按式(3)计算推荐质量 $g_i(\hat{\boldsymbol{r}}_i)$;以评分的数量作为评分成本,即 $c_i(a_i)=|a_i|$;由式(11)和(14)可知,算法的收敛速度与参数 α 的取值密切相关. 考虑到函数 $f(x)=1/\alpha^x$ 在区间[0,70]上的变化趋势,仿真时设 $\alpha=1.2$;分别设 $\mu=0.9$ 和 $\mu=1$ 以模拟如下两种情况:已满足的用户继续提供评分,已满足的用户不再提供评分. 为设置 Γ_i,首先利用 $\boldsymbol{R}$ 和 $\boldsymbol{R}'$ 计算每个用户所能获得的最优推荐质量 $\Gamma_i^{\max}$,然后设 $\Gamma_i=\eta_i\Gamma_i^{\max}$ $(0<\eta_i<1)$. η_i 的设置稍后说明.

4.1.2 MovieLens

本文采用的 MovieLens 数据集包含了 6040 个用户对 3900 部电影的评分. 与上文对 Jester 数据集的处理类似,此处亦假定每个用户体验过的项目的总数为 70,因而我们滤除那些评分数量低于 70 的用户,并滤除那些未获得评分的项目,最终保留 3631 个用户对 3675 个项目的评分,对应的评分矩阵为 $\boldsymbol{R}=[r_{ij}]_{3631\times3675}$,其中 $r_{ij}\in\{0,1,\cdots,5\}$,$r_{ij}=0$ 表示评分未知. 与 Jester 数据不同,这一评分矩阵是非常稀疏的,矩阵中非零元素的比例仅为 6.78%.

均衡学习算法的参数按如下方式设置:矩阵 $\boldsymbol{R}$ 的每一行被视为对应用户的兴趣向量 $\boldsymbol{p}_i$;从 $\boldsymbol{R}$ 的每一行中随机选择 70 个非零元素构成对应用户的完整评分集合,即用户只能对这 70 个元素对应的项目进行评分;按式(2)预测未知评分,设 $|Neighbour(i)|=20$;按式(3)计算推荐质量;参数 α,μ 和 Γ_i 的设置方法与前文所述相同.

4.2 均衡学习仿真结果

为检验算法 1 的可行性,本文测试了多组 $\{\eta_i\}_{i=1}^N$. 给定评分矩阵 $\boldsymbol{R}$ 和参数 μ,分别在如下 4 种设置下运行学习算法:

(1) $\forall i\in\mathcal{N}$, $\eta_i=0.5$;

(2) $\forall i\in\mathcal{N}$, $\eta_i=0.85$;

(3) 随机选择 1% 的用户,令其对应的 $\eta_i=0.85$,其余用户的 $\eta_i=0.5$;

(4) 随机选择 20% 的用户,令其对应的 $\eta_i=0.85$,其余用户的 $\eta_i=0.5$;

为降低随机性的影响,给定一组参数,重复运行算法 5 次. 在每一次运行中,当所有用户都已满足或迭代次数达到 10000 次时,算法终止. 每次运行后,记录算法终止时的迭代次数 n_{stop}、已满足的用户数 N_S 和用户评分完整度的平均值 $\bar{\sigma}_i\triangleq\dfrac{1}{N}\sum_{i=1}^{N}|a_i(n_{\text{stop}})|/|S_i|$.

表 1 和表 2 分别给出了 Jester 数据集和 MovieLens 数据集上的仿真结果. 可以看出:当所有用户对推荐质量有着相似的预期时,即使预期很高($\eta_i=0.85$)并且用户在满足之后不再参与评分,满足均衡也是可以实现的. 给定 μ 的取值,随着用户预期的升高,算法的收敛时间变长,$\bar{\sigma}_i$ 增加,即用户需对更多的项目进行评分才能得到满意的推荐. 给定一组 $\{\eta_i\}_{i=1}^N$,比较不同 μ 对应的仿真结果可以看到,相比于 $\mu=0.9$ 的情况,当 $\mu=1$,算法的收敛时间变长,但用户评分完整度下降. 收敛时间变长的原因在于当 $\mu=0.9$ 时,已满足的用户会继续为推荐质量的提升做出贡献,因此其他那些尚未满足的用户可在短时间内得到满意的推荐. 而当 $\mu=1$ 时,未满足的用户只能依靠他们自己提升推荐质量,因此需要更多的时间才能达到满足均衡. 从评分完整度来看,$\mu=0.9$ 意味着在达到满足均衡时,用户提供的评分数量可能远高于与其预期相对应的必要的评分数量. 而当 $\mu=1$ 时,用户倾向于只提供能满足其个人预期的最少的评分,因此当算法收敛时,用户评分的完整度较低.

当大多数用户对推荐结果有着适中的预期($\eta_i=0.5$)而小部分用户的预期很高时($\eta_i=0.85$),若 $\mu=0.9$,满足均衡仍是可以实现的,只不过此时算法的收敛速度要慢于所有用户预期都不高的情况,并且在达到均衡时用户评分的完整度接近于所有用户都持有高预期的情况. 这一结果表明,为了满足少部分用户的需求,那些预期不高的用户需要在得到满意的推荐之后继续提供很多评分. 当 $\mu=1$ 时,满足的用户不再提供评分. 因而当大多数用户满足之后,余

表 1　Jester 数据集上的均衡学习仿真结果

	runID	$\mu=0.9$					$\mu=1.0$				
		1	2	3	4	5	1	2	3	4	5
$\eta_i=0.5$	n_{stop}	24	28	20	22	26	158	84	130	257	151
	N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
	$\bar{\sigma}_i$	0.531	0.553	0.507	0.519	0.543	0.280	0.281	0.282	0.282	0.282
$\eta_i=0.85$	n_{stop}	1203	1307	1306	1206	1203	8893	10000	9813	8988	9560
	N_S	7200	7200	7200	7200	7200	7200	7199	7200	7200	7200
	$\bar{\sigma}_i$	0.913	0.936	0.931	0.928	0.914	0.781	0.782	0.781	0.782	0.782
1%：$\eta_i=0.85$，99%：$\eta_i=0.5$	n_{stop}	406	404	504	504	403	10000	10000	10000	10000	10000
	N_S	7200	7200	7200	7200	7200	7130	7130	7129	7130	7131
	$\bar{\sigma}_i$	0.906	0.888	0.895	0.896	0.881	0.325	0.326	0.326	0.324	0.324
20%：$\eta_i=0.85$，80%：$\eta_i=0.5$	n_{stop}	804	905	802	902	903	10000	10000	10000	10000	10000
	N_S	7200	7200	7200	7200	7200	6901	6899	6885	6891	6896
	$\bar{\sigma}_i$	0.911	0.920	0.897	0.903	0.910	0.441	0.441	0.441	0.442	0.440

表 2　MovieLens 数据集上的均衡学习仿真结果

	runID	$\mu=0.9$					$\mu=1$				
		1	2	3	4	5	1	2	3	4	5
$\eta_i=0.5$	n_{stop}	27	29	48	28	73	57	28	39	53	37
	N_S	3631	3631	3631	3631	3631	3631	3631	3631	3631	3631
	$\bar{\sigma}_i$	0.814	0.824	0.884	0.821	0.921	0.224	0.224	0.223	0.223	0.226
$\eta_i=0.85$	n_{stop}	528	679	533	494	560	734	610	640	780	571
	N_S	3631	3631	3631	3631	3631	3631	3631	3631	3631	3631
	$\bar{\sigma}_i$	0.990	0.994	0.990	0.989	0.991	0.745	0.744	0.743	0.745	0.745
1%：$\eta_i=0.85$，99%：$\eta_i=0.5$	n_{stop}	118	78	223	68	157	10000	10000	10000	10000	10000
	N_S	3631	3631	3631	3631	3631	3630	3629	3629	3628	3628
	$\bar{\sigma}_i$	0.952	0.928	0.975	0.919	0.964	0.234	0.231	0.229	0.231	0.229
20%：$\eta_i=0.85$，80%：$\eta_i=0.5$	n_{stop}	482	458	302	462	678	10000	10000	10000	10000	10000
	N_S	3631	3631	3631	3631	3631	3624	3619	3625	3622	3624
	$\bar{\sigma}_i$	0.989	0.989	0.981	0.989	0.993	0.350	0.349	0.350	0.349	0.350

下那些未满足的用户很难再使推荐质量有显著提升，算法无法在 10000 次迭代内收敛. 观察 $\bar{\sigma}_i$ 的取值可以看出，大多数用户只是提供了能够满足其中等预期的评分，但这一数量的评分不足以产生很高质量的推荐.

为更清楚地观察用户预期对均衡学习结果的影响，我们将不同参数设置对应的 $|\mathcal{N}_S(n)|$ 变化曲线绘制于图 5 中（以 Jester 数据集为例）. 可以看到，在第 4 种设置下（图 5 中圆圈标记的曲线），经过约 15 次迭代，80%的用户就已满足，这与第 1 种设置的仿真结果（图 5 中用圆点标记的曲线）类似. 而在此之后，$|\mathcal{N}_S(n)|$ 的增长明显变缓，经过很多次迭代才达到满足均衡.

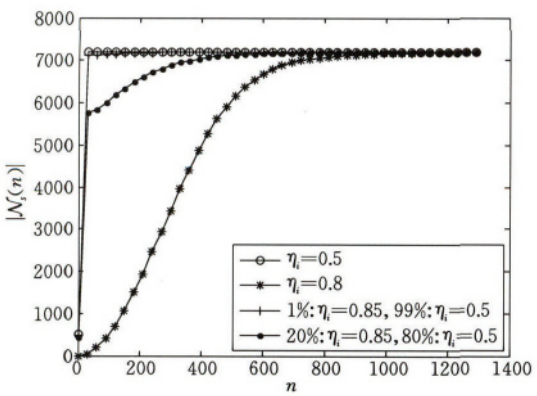


图 5　均衡学习过程中“满足”用户的数量的变化（Jester 数据集，$\mu=0.9$）

由上述仿真结果可以得到关于协同过滤系统中满足均衡的直观理解：若所有用户对推荐质量的预期都不高，则低成本的满足均衡是可以实现的，即每个用户只需提供较少的评分就能得到满意的推荐. 稍后将通过另一组仿真对 3.2 节提出的收敛条件进行验证.

4.3　奖励机制

从上一小节给出的仿真结果可以看出，当不同用户对推荐质量有着相似的预期时，仅靠推荐质量这一内部激励就可以让用户自发地提供足够多的评分数据以使推荐服务器产生符合所有用户预期的推荐. 而当不同用户对推荐质量的预期相差较大时且用户在满足之后不再对更多的项目评分时，均衡学习算法无法收敛到满足均衡，这意味着此时推荐服务器有必要提供一些外部激励以促使用户参与评分.

假定当一个用户选择行动 $A^{(k)}\in\mathcal{A}$ 时，用户付出的成本为 $c_i(A^{(k)})\triangleq|A^{(k)}|$，推荐服务器给予用户的奖励为 $b(A^{(k)})\triangleq\kappa|A^{(k)}|$，其中参数 κ 表示用户对一个项目评分后所能获得的奖励（$0<\kappa<1$）. 由算法 1 可知，用户在选择行动时倾向于选择低成本的行

动.推荐服务器向用户给予奖励,这相当于用户的评分成本由原来的 $c_i(A^{(k)})$ 降低为 $c_i(A^{(k)})-b(A^{(k)})$,因而用户选择较多项目进行评分的概率变高.另一方面,当用户的预期得到满足之后,虽然推荐质量这一内在激励失效,但如果推荐服务器提供奖励,用户仍有动机对更多的项目评分.本文规定,在均衡学习过程中,若存在评分奖励,已满足的用户维持原有行动的概率为 $\mu\triangleq 1-\kappa$.由 $0<\kappa<1$ 可知,此时 $\mu<1$,因而均衡学习算法总能收敛到满足均衡.

本文在 Jester 数据集上对带有评分奖励的均衡学习算法进行了仿真.在仿真中,分别设 $\kappa=0.01$,0.1,0.5,0.9,其他参数的设计与上一小节相同.仿真结果如表 3 所示.对比表 1 和表 3 可以看出,当推荐服务器实施评分奖励时,均衡学习算法的收敛速度明显加快.奖励越高,算法的收敛速度越快.例如,当仅有 1%的用户对推荐质量的预期很高时,由表 1 可知,若没有评分奖励且 $\mu=0.9$,学习算法要经过至少 400 次迭代才能达到均衡;而当推荐服务器提供评分奖励且 $\kappa=0.1$ 时,此时亦有 $\mu=0.9$,但学习算法只需经过约 200 次迭代便可达到均衡.

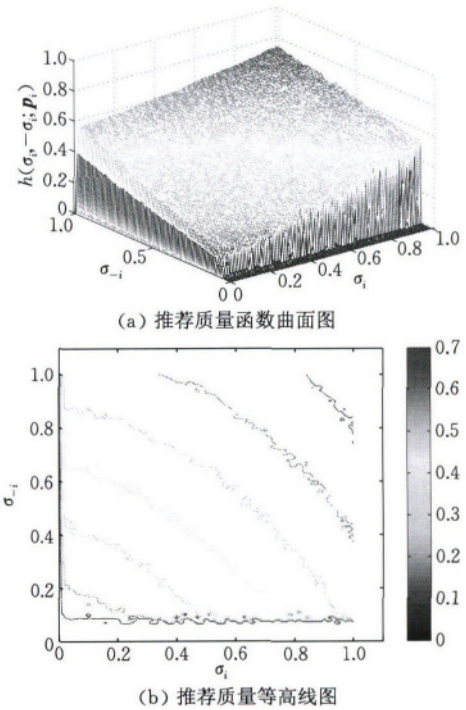
表 3 不同奖励下的均衡学习仿真结果(Jester 数据集)

	runID	1%:$\eta_i=0.85$, 99%:$\eta_i=0.5$					20%:$\eta_i=0.85$, 80%:$\eta_i=0.5$				
		1	2	3	4	5	1	2	3	4	5
$\kappa=0.01$	n_{stop}	102	172	184	201	102	401	302	303	304	304
	N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
	$\bar{\sigma}_i$	0.939	0.888	0.891	0.935	0.938	0.960	0.964	0.973	0.980	0.979
$\kappa=0.1$	n_{stop}	183	205	201	201	202	301	302	304	303	302
	N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
	$\bar{\sigma}_i$	0.859	0.931	0.885	0.886	0.899	0.904	0.914	0.931	0.923	0.914
$\kappa=0.5$	n_{stop}	15	19	16	16	18	21	18	20	20	21
	N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
	$\bar{\sigma}_i$	0.884	0.907	0.889	0.890	0.904	0.916	0.903	0.911	0.912	0.915
$\kappa=0.9$	n_{stop}	3	3	3	3	3	4	4	4	5	4
	N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
	$\bar{\sigma}_i$	0.925	0.926	0.922	0.923	0.925	0.962	0.963	0.963	0.982	0.963

上述关于奖励机制的简单讨论表明,推荐服务器可通过向用户提供外部奖励的方式促使用户间的博弈尽快达到满足均衡.推荐服务器可通过观察用户以往的评分行为大致估计用户的评分成本、用户对推荐质量的预期等,进而设计合适的奖励规则.具体的设计方法有待进一步研究.

4.4 推荐质量和评分完整度之间的关系

第 3.2 节给出的关于均衡学习算法收敛性的理论分析是基于若干假设的.在检验收敛条件之前,此处先利用 Jester 数据集对假设 1 和假设 2 的合理性进行说明.实验方法如下:利用评分矩阵 $\boldsymbol{R}'$ 为每个用户 i 构造一组矩阵 $\{\boldsymbol{R}_{i,k}\}$,其中每个矩阵 $\boldsymbol{R}_{i,k}$ 对应于一组特定的 σ_i 和 σ_{-i}.σ_i 在集合 $\left\{\frac{1}{70},\frac{2}{70},\cdots,\frac{70}{70}\right\}$ 中取值,σ_{-i} 在集合 $\left\{\frac{1}{100},\frac{2}{100},\cdots,\frac{100}{100}\right\}$ 中取值.例如,给定 $\sigma_i=\frac{5}{70}$ 和 $\sigma_{-i}=\frac{10}{100}$,先从矩阵 $\boldsymbol{R}'$ 的第 i 行中随机选择 5 个非零元素,然后将这些元素置为零,接着再从 $\boldsymbol{R}'$ 的其他行中随机选择 90%的非零元素,并将这些元素置为零.将协同过滤算法应用每个 $\boldsymbol{R}_{i,k}$,并根据用户 i 的兴趣向量对推荐结果进行评估,得到与每组 (σ_i,σ_{-i}) 对应的 $h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)$.根据每个用户的实验数据 $\{(\sigma_i,\sigma_{-i},h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i))\}$,可以绘制出与该用户对应的 $h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)$ 曲面图.图 6(a)给出了某一用户对应的曲面图,其他用户的曲面图与之类似.从图 6(a)可以看到,推荐质量的确会随 σ_i 和 σ_{-i}



(a) 推荐质量函数曲面图

(b) 推荐质量等高线图

图 6 推荐质量与评分完整度的关系

的增加而提升，这表明假设 1 是合理的. 另外，从图 6(b)所示的等高线图可以看出，给定 $h(\sigma_i,\sigma_{-i};\boldsymbol{p}_i)$ 的取值，σ_i 和 σ_{-i} 之间的关系可大致用二次曲线描述，这表明假设 2 也是合理的.

4.5　收敛条件测试

本文利用 Jester 数据集对 3.2 节给出的均衡学习算法收敛条件进行了验证. 按照假设 2 的描述，全部 7200 个用户被随机分为 N_A 和 N_B 两组，η_A 设为 0.5，ρ_N 分别设为 0.01、0.10 和 0.20. 给定一组 η_A 和 ρ_N，先按式(26)计算 θ_B，然后将 η_B 分别设为 $\eta_B=\theta_B-0.05$，$\eta_B=\theta_B$ 和 $\eta_B=\theta_B+0.05$. 为证明在所有用户都有高预期的情况下算法是可以收敛的，对 $\eta_A=\eta_B=\theta_B+0.05$ 的情况也进行了仿真. 给定一组(ρ_N，η_A，η_B)，将简化的均衡学习算法运行 10 次. 在每一次运行中，当出现如下两种情况之一时算法终止：所有用户均已满足，未满足的用户已经对其体验过的所有项目进行了评分. 算法终止时的迭代次数记为 n_{stop}，满足的用户数记为 N_S.

从表 4 所示的仿真结果可以看到，给定 η_A 和 ρ_N，当 $\eta_B=\theta_B-0.05$ 时，满足均衡总是可以实现的；当 $\eta_B=\theta_B$ 时，$\mathcal{N}_B$ 中的大部分用户可以满足；当 $\eta_B=\theta_B+0.05$ 时，$\mathcal{N}_B$ 中的绝大多数用户无法满足. 上述结果与 3.2 节的理论分析结果有些许出入，因为由式(26)可知，当 $\eta_B\leq\theta_B$ 时满足均衡应是可以实现的. 造成理论分析与仿真结果不一致的原因可能是用户评分完整度与推荐质量之间的关系并不严格符合假设 2 所定义的二次曲线关系. 从与 $\eta_A=\eta_B=\theta_B+0.05$ 对应的结果可看出，即使所有用户都有较高的预期，用户不提供完整的评分也能实现满足均衡($n_{\text{stop}}<|S_i|$). 这一结果进一步表明，在一个协同过滤系统中，用户能否通过自发的评分实现满足均衡，取决于不同用户对推荐质量是否有着相似的预期.

表 4　收敛性测试结果

ρ_N	η_A	θ_B	η_B	runID	1	2	3	4	5	6	7	8	9	10
0.01	0.5	0.792	0.742	n_{stop}	68	67	68	67	68	67	68	68	68	67
				N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
			0.792	n_{stop}	70	70	70	70	70	70	70	70	70	70
				N_S	7178	7178	7177	7176	7176	7179	7176	7180	7176	7177
			0.842	n_{stop}	70	70	70	70	70	70	70	70	70	70
				N_S	7130	7130	7131	7130	7129	7129	7130	7129	7129	7129
	$\eta_A=\eta_B=0.842$			n_{stop}	65	65	65	65	65	65	66	64	66	66
				N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
0.1	0.5	0.807	0.757	n_{stop}	69	68	68	68	69	69	69	68	69	68
				N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
			0.807	n_{stop}	70	70	70	70	70	70	70	70	70	70
				N_S	7132	7130	7139	7141	7141	7131	7138	7140	7124	7137
			0.857	n_{stop}	70	70	70	70	70	70	70	70	70	70
				N_S	6642	6618	6627	6630	6619	6627	6630	6640	6629	6626
	$\eta_A=\eta_B=0.857$			n_{stop}	66	66	65	66	66	66	67	67	66	66
				N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
0.2	0.5	0.824	0.775	n_{stop}	69	68	68	69	69	69	68	68	69	69
				N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
			0.825	n_{stop}	70	70	70	70	70	70	70	70	70	70
				N_S	7133	7146	7139	7142	7130	7130	7148	7134	7137	7136
			0.875	n_{stop}	70	70	70	70	70	70	70	70	70	70
				N_S	6368	6400	6383	6398	6387	6409	6363	6405	6400	6419
	$\eta_A=\eta_B=0.875$			n_{stop}	66	67	67	66	67	66	66	67	67	67
				N_S	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200

5　结　论

在协同过滤推荐系统中，用户的评分行为是相互影响的，各用户以推荐服务器为媒介进行着某种复杂的交互. 本文将用户间的这种交互建模为一种不完全信息博弈，定义了该博弈对应的满足均衡，并提出了一种均衡学习算法. 所提算法的基本思想是：用户在与推荐服务器迭代交互的过程中，根据当前所得推荐是否达到预期来决定是否提供更多的评分. 在适当的简化假设下，本文分析了算法的收敛条件. 在真实数据上的仿真结果表明，当所有用户对推荐质量有着相似的预期时，通过用户自发的评分行为确实是可以实现满足均衡的.

用户的积极参与对协同过滤系统的良好运作至关重要. 本文将推荐质量视为促使用户参与评分的一种内部激励，但给出的博弈分析也可为外部激励机制的设计提供一些启发. 在接下来的工作中，我们

将研究如何结合内部激励和外部激励来引导用户在协作式系统中的行为.

参 考 文 献

[1] Ma Hong-Wei, Zhang Guang-Wei, Li Peng. Survey of collaborative filtering algorithms. Journal of Chinese Computer Systems, 2009, 30(7): 1282-1288(in Chinese)
(马宏伟, 张光卫, 李鹏. 协同过滤推荐算法综述. 小型微型计算机系统, 2009, 30(7): 1282-1288)

[2] Su X, Dhaliwal S. Incentive mechanisms in P2P media streaming systems. IEEE Internet Computing, 2010, 14(5): 1089-7801

[3] Wu W, Ma R T B, Lui J C S. Distributed caching via rewarding: An incentive scheme design in P2P-vod systems. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(3): 612-621

[4] Lee J S, Hoh B. Sell your experiences: A market mechanism based incentive for participatory sensing//Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications. Mannheim, Germany, 2010: 60-68

[5] Jaimes L G, Vergara-Laurens I, Labrador M A. A location-based incentive mechanism for participatory sensing systems with budget constraints//Proceedings of the 10th Annual IEEE International Conference on Pervasive Computing and Communications. Lugano, Switzerland, 2012: 103-108

[6] Kang Lin, Li Xiu-Hua, Wang Wei-Dong. A survey on incentive mechanism for participatory sensing systems. Microcontrollers & Embedded Systems, 2013, 13(8): 9-12 (in Chinese)
(康琳, 李秀华, 王卫东. 参与感知世界的激励机制研究. 单片与嵌入式系统应用, 2013, 13(8): 9-12)

[7] Gao Y, Chen Y, Liu K. On cost-effective incentive mechanisms in microtask crowdsourcing. IEEE Transactions on Computational Intelligence and AI in Games, 2015, 7(1): 3-15

[8] Luo Yun-Feng. Game Theory. Beijing: Tsinghua University Press, 2007(in Chinese)
(罗云峰. 博弈论教程. 北京: 清华大学出版社, 2007)

[9] Halkidi M, Koutsopoulos I. A game theoretic framework for data privacy preservation in recommender systems//Gunopulos D, Hofmann T, Malerba D, Vazirgiannis M eds. Machine Learning and Knowledge Discovery in Databases. Berlin: Springer Berlin Heidelberg, 2011: 629-644

[10] Ross S, Chaib-draa B. Satisfaction equilibrium: Achieving cooperation in incomplete information games//Lamontagne L, Marchand M eds. Advances in Artificial Intelligence. Berlin: Springer Berlin Heidelberg, 2006: 61-72

[11] Perlaza S M, Tembine H, Lasaulce S, et al. Quality-of-service provisioning in decentralized networks: A satisfaction equilibrium approach. IEEE Journal of Selected Topics in Signal Processing, 2012, 6(2): 104-116

[12] Perlaza S M, Poor H V, Han Z. Learning efficient satisfaction equilibria via trial and error//Proceedings of the 46th Asilomar Conference on Signals, Systems and Computers. Pacific Grove, USA, 2012: 676-680

[13] Mériaux F, Perlaza S, Lasaulce S, et al. Achievability of efficient satisfaction equilibria in self-configuring networks. Game Theory for Networks. Berlin: Springer Berlin Heidelberg, 2012: 1-15

[14] Marden J R, Young H P, Pao L Y. Achieving pareto optimality through distributed learning//Proceedings of the 51st Annual Conference on Decision and Control. Maui, Hawaii, 2012: 7419-7424

[15] Goldberg K, Roeder T, Gupta D, et al. Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval, 2001, 4(2): 133-151

XU Lei, born in 1986, Ph. D. Her research interests include information sharing and application of game theory.

YANG Cheng, born in 1974, Ph. D., professor. His research interest is information security.

JIANG Chun-Xiao, born in 1987, Ph. D. His research interests include information sharing and application of game theory.

REN Yong, born in 1963, Ph. D., professor. His research interests include information sharing and complex system.

Background

The predominant approaching to building recommendation systems is collaborative filtering (CF). The basic idea of CF is to utilize the ratings provided by users to match users with similar interests. User participation is of vital importance for the success of CF. To encourage user participation, the recommendation server can offer incentives (e. g. monetary rewards) to users. Although mechanisms proposed particularly for CF systems are rare, incentive mechanisms have been extensively studied in similar contexts including peer-to-peer resource sharing, participatory sensing, crowdsourcing, etc.

Besides external incentives, user participation can also be motivated by intrinsic factors. That meas a user may get better recommendations if he/she reveals more information about his/her preferences by rating more items. However, rating more items means the user has to pay more cost (e. g. time cost). Moreover, as the name collaborative filtering suggests, whether a user can get accurate recommendations depends not only on the ratings provided by the user himself/herself, but also on the ratings provided by others. Furthermore, users are usually rational, in the sense that a user wishes to obtain good recommendations by rating only a few items. In such a case, it is natural to employ game theory to model the interactions among users in a CF system.

In this paper, we build a game-theoretic model to study users' rating behaviors in a CF-based recommendation system. Application of game theory has been seen in a few studies of user behavior in a context where individuals' behaviors affect each other. Different from previous study, we model the interactions among users as a satisfactory game with incomplete information, i. e., each user only has the knowledge of his/her own ratings and recommendations, while others' ratings cannot be observed. Meanwhile, the CF algorithm adopted by the RS is also unknown to users. To analyze the game with incomplete information, we apply the notion of satisfaction equilibrium (SE). To the best of our knowledge, this is the first time that SE is applied to study CF. Based on the characteristics of recommendation systems, we design a learning algorithm which allows users to achieve a SE. Convergence of the proposed learning algorithm is analyzed theoretically and experimentally.

The game-theoretic analysis we presented in this paper may provide some implications to the study of user behaviors in collaborative systems. The derived convergence conditions may also be helpful to the design of incentive mechanisms.