Addressing the Item Cold-Start Problem by Attribute-Driven Active Learning

Yu Zhu, Jinghao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai, *Member, IEEE*

Abstract—In recommender systems, cold-start issues are situations where no previous events, e.g., ratings, are known for certain users or items. In this paper, we focus on the item cold-start problem. Both content information (e.g., item attributes) and initial user ratings are valuable for seizing users' preferences on a new item. However, previous methods for the item cold-start problem either (1) incorporate content information into collaborative filtering to perform hybrid recommendation, or (2) actively select users to rate the new item without considering content information and then do collaborative filtering. In this paper, we propose a novel recommendation scheme for the item cold-start problem by leveraging both active learning and items' attribute information. Specifically, we design useful user selection criteria based on items' attributes and users' rating history, and combine the criteria in an optimization framework for selecting users. By exploiting the feedback ratings, users' previous ratings and items' attributes, we then generate accurate rating predictions for the other unselected users. Experimental results on two real-world datasets show the superiority of our proposed method over traditional methods.

Index Terms—Recommender systems, active learning, cold-start, exploitation and exploration

1 Introduction

Recommender systems (RS) have become extremely common in recent years, and are applied in a variety of domains, from virtual community websites like movielens.org to electronic commerce companies like amazon.com. In spite of the widespread applications of RS, one difficult and common problem is the cold-start problem, where no prior events, like ratings or clicks, are known for certain users or items. The user cold-start problem may lead to the loss of new users due to the low accuracy of recommendations in the early stage. The item cold-start problem may make the new item miss the opportunity to be recommended and remain "cold" all the time. In this paper, we focus on the item cold-start problem, where recommendations are required for items that no one has yet rated.

Content information, such as item attributes, were exploited to address such issues in previous methods [1], [2], [3]. However, items with similar attributes may be of different interest for the same user. As shown in Fig. 1 (data is collected from *IMDB*[1]), movie *Taken* is favored by many people after release, with the mean rating equal to 8.0. When the follow-up *Taken 3* was first released in 2014, it can be seen as a "cold" film. Since genres, screenwriters and many actors of these two films are the same, then if we exploit film attributes to perform hybrid recommendations, we may recommend this "cold" film to users who favored *Taken* before. However, as can be seen in the figure, the peak of *Taken 3*'s overall ratings moves down to rating 6, which means that many users might favor *Taken* but would give low ratings to *Taken 3*, i.e., the recommendation of *Taken 3* to users who favored *Taken* before might be wrong. Therefore, it is not a safe way to handle the cold-start issue based on film attributes only. A natural solution is to select a small set of users to watch this "cold" film first, whose feedback can give us more understanding of users' preferences on this film. Then we can perform more accurate recommendations. Actually, this is the key idea of active learning in the machine learning literature [4].

Most works that apply active learning to recommender systems focus on the user cold-start problem [5], [6], [7]. New users' preferences are typically obtained by (1) directly interviewing the user about what his interest is or (2) asking him to rate several items from carefully constructed seed sets (items in these seed sets would be shown to every new user). Seed sets may be constructed based on popularity, contention and coverage [6]. However, the item cold-start

- *Y. Zhu is with the State Key Laboratory of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou 310027, China, and the Alibaba Group, Hangzhou 311100, China. E-mail: zy143829@alibaba-inc.com.*
- *J. Lin and S. He are with the State Key Laboratory of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou 310027, China. E-mail: fenixl@zju.edu.cn, frankheshibi@gmail.com.*
- *B. Wang is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. E-mail: beidouw@sfu.ca.*
- *Z. Guan is with the College of Information and Technology, Northwest University of China, Xian 710127, China. E-mail: ziyuguan@nwu.edu.cn.*
- *H. Liu is with the College of Computer Science, Zhejiang University, Hangzhou 310027, China. E-mail: haifengliu@zju.edu.cn.*
- *D. Cai is with the State Key Laboratory of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou 310027, China, and the Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Hangzhou 310027, China. E-mail: dcai@zju.edu.cn.*

Manuscript received 17 May 2016; revised 19 Nov. 2018; accepted 2 Jan. 2019. Date of publication 9 Jan. 2019; date of current version 5 Mar. 2020. (Corresponding author: Deng Cai.)
Recommended for acceptance by X. Lin.
Digital Object Identifier no. 10.1109/TKDE.2019.2891530

1. http://www.imdb.com/

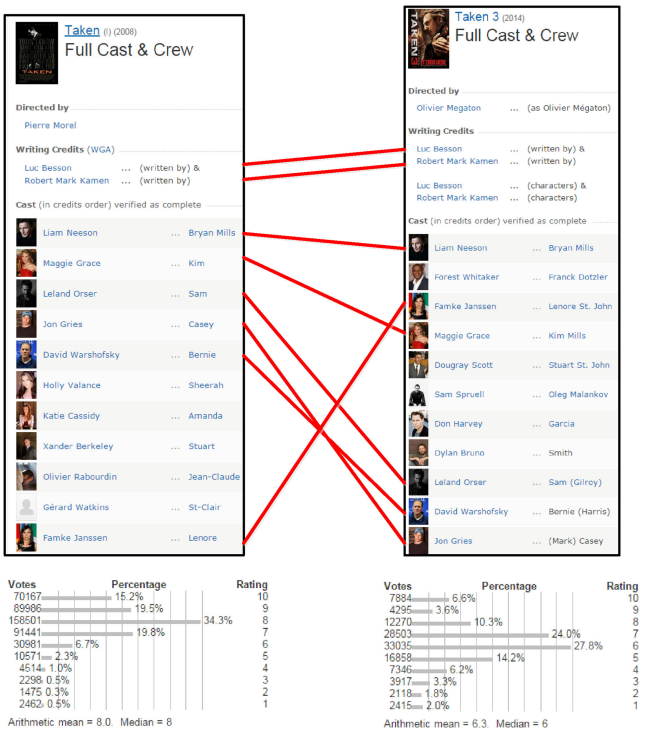


Fig. 1. Attribute information and users' rating distributions of films *Taken* and *Taken 3*. Genres of these two films are both *Action* and *Thriller*. Common scriptwriters and actors are connected by red lines. The overall ratings of *Taken* are high with the mean equal to 8.0 and the mean rating is 6.3 for *Taken 3*. Attributes are modeled as features in this paper, whose values are 1 if the corresponding attributes exist or 0 otherwise. In this example, features include all genres, directors, scriptwriters and actors.

problem is different because (1) items cannot be interviewed and (2) typically no users are willing to rate every new item. For the latter issue, a better way is to construct different user sets to rate different new items, ensuring that users are not always selected for rating requests. In addition, for each new item, the user set should be carefully constructed so that we can learn this item as much as possible. However, limited works have been conducted to address the item cold-start problem by active learning. [8], [9] adopt the active learning idea but ignore items' attribute information. Actually, the new item's attributes give us some understanding of this item and can be exploited to improve our user selection strategy. For example, we tend to select users who favor attributes existing in the new item, since these users are more willing to give ratings.

In this paper, we propose a novel recommendation framework for the item cold-start problem, where items' attributes are exploited to improve active learning methods in recommender systems. The attribute-driven active learning scheme has the following characteristics:

- Explicitly distinguishing (1) whether a user will rate the new item and (2) what rating the user will give to the new item. The former helps us to select users who are willing to give feedback ratings. The latter allows us to exploit the rating distribution to improve the selection strategy. For example, we expect to select users who give diverse ratings to generate unbiased predictions. This is easy to understand since if we select users who all give high ratings, then the trained prediction model will generate high biased ratings for all other users, though the other users may not favor the new item at all.
- Personalized selection strategy to ensure fairness. Our selection strategy is based on four criteria, of which two are personalized. The personalized criteria ensure that for new items with different attributes, users selected by our method would be different. This can avoid selecting the same user to rate every new item, which will negatively influence the user experience. These criteria are uniformly modeled as an integer quadratic programming (IQP) problem, which can be efficiently solved by some relaxation.
- Dynamic active learning budget. In previous active learning works [9], [10], the *budget* (i.e., the number of users selected for rating requests) for a new item is fixed. However, in real-world applications, (1) some new items are under the attention of a small set of users (not popular), e.g., films with unpopular actors and directors, and (2) some would be obviously favored by almost all users (popular and not controversial), e.g., *Harry Potter and the Deathly Hallows: Part 2*,[2] while (3) others are popular but controversial, and the recommender is not sure about users' preferences on them, e.g., although *Taken 3* is famous, the qualities of films previously acted by its main actors vary a lot, thus it is difficult to predict users' preferences on *Taken 3*. It is the items in the third case that need more feedback ratings so as to be learned more about. In this paper, we are the first to propose a dynamic active learning budget so that the limited active learning resources will be properly distributed, which can improve the overall prediction accuracy.
- Considering *exploitation*, *exploration* and their trade-off. Traditional active learning methods aim at maximizing the performance measured in the prediction phase [11], [12], regardless of the active learning phase, since they assume the labeling cost for each instance is the same. However, in our active learning phase, we prefer to select a user who is willing to give the rating rather than a user who is not, because the latter will negatively influence the user experience. Our solutions are inspired by [13], [14], which try to maximize the sum of *rewards* by balancing the trade-off of *exploitation* and *exploration*. The *rewards* in our task contain two parts, i.e., the user experience in the active learning phase and the prediction phase, respectively. By exploiting "existing knowledge" (*exploitation*) from the model trained in Fig. 3b, we can select willing users to obtain good user experience in the active learning phase. For the user experience in the prediction phase, we want to learn as much "new knowledge" (*exploration*) about unselected users' preferences as possible, so as to generate accurate rating predictions for them. Our method simultaneously considers *exploitation* and *exploration*, and can further balance their trade-off by adjusting the parameter setting.

2. http://www.imdb.com/title/tt1201607/

The rest of our paper is organized as follows: Section 2 surveys the related work. We introduce the task definition and solution overview in Section 3. Next, we present the details of our proposed method in Section 4. A variety of experimental results are presented in Section 5. Finally, we provide some concluding remarks and suggestions for future work in Section 6.

2 Related Work

2.1 The Item Cold-Start Problem

To address the item cold-start problem, a common solution is to perform hybrid recommendations by combining content information and collaborative filtering [15], [16], [17], [18]. A regression-based latent factor model is proposed in [15] to address both cold and warm item recommendations in the presence of items' features. Items' latent factors are obtained by low-rank matrix decomposition. [17] solves a convex optimization problem, instead of the matrix decomposition, to improve this work. Another approach based on Boltzmann machines is proposed in [16], [19] to solve the item cold-start problem, which also combines content and collaborative information. LCE [20] exploits the manifold structure of the data to improve the performance of hybrid recommendations. Other works are under a different setting where few ratings of new items exist, but no items' attribute information is known. [21], [22] use a linear combination of raters' latent factors weighted by their ratings to estimate new items' latent factors.

2.2 Active Learning in Recommender Systems

Most active learning methods in recommender systems focus on the user cold-start problem, where they select items to be rated by newly-signed users [4], [23]. We briefly introduce these methods since most of them can also be adapted to our task. The Popularity strategy [7], [24] and the Coverage strategy [24] are two representative attention-based methods, where the former selects items that have been frequently rated by users and the latter selects items that have been highly co-rated with other items. Uncertainty reduction methods aim at reducing the uncertainty of rating estimates [13], [14], [24], model parameters [25], [26] and decision boundaries [27]. Error reduction methods try to reduce the prediction error on the testing set by either (1) optimizing the performance measure (e.g., minimizing *RMSE*) on the training set [7], [24], or (2) directly controlling the factors that influence the prediction error on the testing set [28], [29]. [30] uses some initial ratings to perform personalized active learning in a non-attribute context. There are also combined strategies [13], [31], [32] considering several objectives at the same time. When applied to our task, some of these works require a few initial ratings on new items, which are not available in our setting. The other works do not need initial ratings, but perform active learning regardless of the content information. However, the new item's content information gives us some understanding of the new item and we can exploit it to better perform active learning. In addition, methods such as the Popularity strategy [7], [24] and the Coverage strategy [24] always select the same set of users, which negatively influence the user experience.

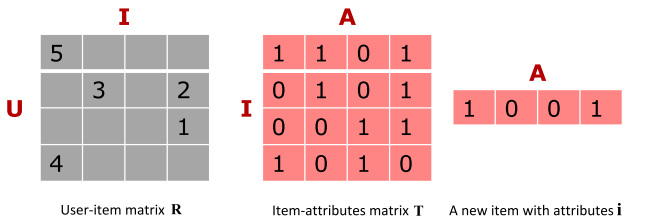


Fig. 2. Representations of user-item matrix $\mathbf{R} \in R^{|U|\times|I|}$, item-attributes matrix $\mathbf{T} \in R^{|I|\times|A|}$, and the vector $\mathbf{i} \in R^{1\times|A|}$ representing a new item with attributes.

Aharon et al. [8], [9], [33], [34] are works which also address the item cold-start problem in an active learning scheme. However, they all focus on the pure collaborative filtering model and do not consider the content information either.

2.3 The Exploitation-Exploration Trade-Off

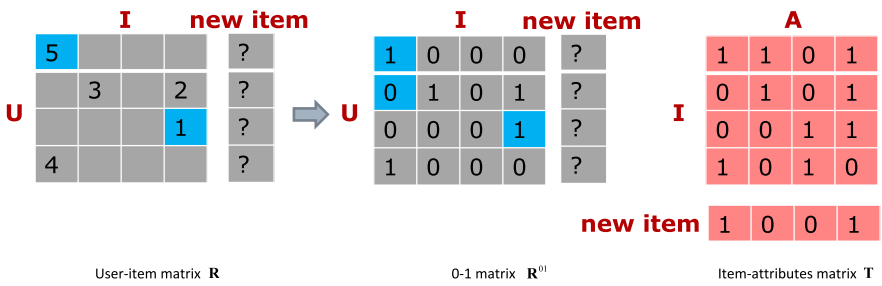
Some works also consider the exploitation-exploration trade-off [13], [14]. Many of the promising solutions come from the study of the multi-armed bandit problem [35]. The key idea of these solutions is to simultaneously optimize one's decisions based on existing knowledge (i.e., *exploitation*) and new knowledge which would be acquired through these decisions (i.e., *exploration*), in order to maximize the sum of rewards earned through a sequence of actions. The ϵ-Greedy algorithm [36] selects the arm which has the best estimated mean reward with probability $1-\epsilon$, and otherwise randomly selects another arm. UCB-like (UCB refers to Upper Confidence Bound) algorithms [37], [38], [39] first calculate the confidence bound of all arms and then select the arm with the largest upper confidence bound. The insight is that arms with a large mean reward (exploitation) and high uncertainty (exploration) would have large upper confidence bound. Thompson Sampling algorithms [40], [41], [42] first calculate the probability distribution of the mean reward for each arm, then draw a value from each distribution and finally select the arm with the largest drawn value.

Our task and the multi-armed bandit problem share some common features, e.g., both considering the exploitation-exploration trade-off. However, they have some key differences. In the multi-armed bandit problem, the arms are selected one by one and a reward is generated immediately after each arm is selected. Hence, many solutions (e.g., UCB-like algorithms, Thompson Sampling algorithms) design their selecting strategies based on previous rewards. However, in our setting, a batch of users are selected at the same time, without knowing other users' feedback (reward), thus many solutions for the multi-armed bandit problem cannot be applied to our task.

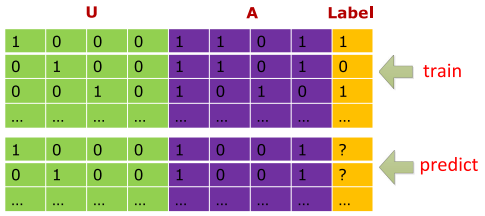
3 Preliminaries

3.1 Task Definition and Solution Overview

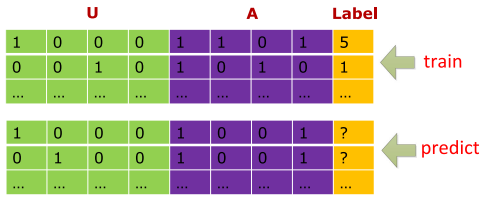
We use U, I and A to denote the users, items and attributes set, respectively. Our task is that given a user-item rating matrix $\mathbf{R} \in R^{|U|\times|I|}$, an item-attribute matrix $\mathbf{T} \in R^{|I|\times|A|}$ and a new item i_{new}, whose attributes are denoted as a vector $\mathbf{i} \in R^{1\times|A|}$, we want to predict users' ratings on the new item $predict\_rating(u, i_{new}), u \in U$. $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{i}$ are shown in Fig. 2. In this paper, the task is solved via two phases. The first one



(a) Representations of $\mathbf{R}$, $\mathbf{R}^{01}$ and $\mathbf{T}$



(b) Factorization Machines modeling whether users will rate items



(c) Factorization Machines modeling what ratings users will give to items

Fig. 3. (a) Shows how $\mathbf{R}$ is transformed to $\mathbf{R}^{01}$ [45]. (b) Is the classification model. Each row is an instance, which corresponds to an entry of $\mathbf{R}^{01}$. For example, the first row corresponds to $\mathbf{R}^{01}$'s entry located in row 1 and column 1. Row 1 indicates the first user, thus the feature set labeled as 'U' is (1,0,0,0) (one-hot representation). Column 1 indicates the first item, thus the feature set labeled as 'A' is equal to the first row of $\mathbf{T}$, i.e., attributes of the first item. For testing, each user is predicted to show whether he will rate the new item. (c) Is the regression model. Each row is an instance, which corresponds to an entry of $\mathbf{R}$. For testing, each user is predicted to show what rating he will give to the new item.

is the active learning phase, which is to solve which users should be selected to rate i_{new}, so as to learn i_{new} as much as possible. The second one is the prediction phase, which is to solve once given selected users' feedback, how to accurately predict unselected users' ratings on i_{new}. For the active learning phase, we carefully select users based on four criteria, which involve both classification and regression tasks. For the prediction phase, it is a pure regression task. We use Factorization Machines [43] to model these classification and regression tasks, since it has been proved to perform excellently in many recommendation applications [1], [43], [44].

3.2 Factorization Machines and Its Adaptation to Our Task

Factorization Machines (FM) [43] is a state-of-the-art framework for latent factor models which can incorporate rich features. Here we briefly introduce it. The prediction problem is described by a matrix $\mathbf{X} \in R^{N \times D}$ and a vector $\mathbf{y} \in R^{N \times 1}$, where each row $\mathbf{x} \in R^{1 \times D}$ of $\mathbf{X}$ is one instance with D real-valued features and each entry y in $\mathbf{y}$ is the corresponding label. FM can model nested feature interactions up to an arbitrary order. For feature interactions up to 2-order, they are modeled as

$$y'(\mathbf{x}) = w(0) + \sum_{i=1}^{D} w(i)x(i) + \sum_{i=1}^{D}\sum_{j=i+1}^{D} x(i)x(j)\sum_{f=1}^{k} V(i,f)V(j,f), \quad (1)$$

where k is the dimensionality of the factorization and the model parameters are: $w(0) \in R, \mathbf{w} \in R^{D \times 1}, \mathbf{V} \in R^{D \times k}$. $w(i)$, $x(i)$ and $V(i,f)$ are entries of $\mathbf{w}$, $\mathbf{x}$ and $\mathbf{V}$, respectively.

We adapt FM to our regression task (i.e., predict what rating a user will give to an item) and classification task (i.e., predict whether a user will rate an item) as follows. For the regression task, features include users, items and attributes of items, and labels are ratings. The first term of the right-hand side in Eq. (1) is a bias of the system. If $w(0)$ is large, then there is a bias towards high ratings. The second term is a bias of unary features and the last term is a bias of feature interactions. For the classification task, the analysis of each term in Eq. (1) is similar, except that labels now represent whether users will rate items or not.

4 Our Method

In this section, we describe our method in detail. For the active learning phase, we first introduce how we select users to rate i_{new} based on four criteria in Section 4.1, then describe how the dynamic active learning budget improves our selection strategy in Section 4.2. Next, the prediction phase is presented in Section 4.3. Finally, we analyze the superiority of our method from the exploitation-exploration perspective in Section 4.4.

4.1 Select Users to Rate the New Item

Users are selected based on the following four criteria.

Criterion (1): Selected Users are with High Possibility to Rate $\mathbf{i_{new}}$. We first transform the user-item rating matrix $\mathbf{R}$ to a 0-1 matrix $\mathbf{R}^{01}$ [45], where all entries with ratings are assigned to 1, and the other entries are assigned to 0 (see Fig. 3a). Then we use FM [43] to model it as a classification task, where all entries equal to 1 are positive instances, and a same number of negative instances are sampled from the other entries. Features contain users and attributes (without items). The model is trained based on $\mathbf{R}^{01}$ and $\mathbf{T}$. The general process is shown in Fig. 3b. Finally, a vector $\mathbf{p} \in R^{|U| \times 1}$ is defined as follows:

$$p(m) = willing\_score(u_m, i_{new}), u_m \in U, \tag{2}$$

where $willing\_score(u_m, i_{new})$ is the possibility that user u_m will rate i_{new}, which is predicted by our learned classification model. We tend to select u_m if $p(m)$ is large.

Criterion (2): Selected Users' Potential Ratings are Diverse. *Potential ratings* are users' ratings on i_{new} purely estimated according to i_{new}'s attributes (without feedback since there is no feedback yet). We expect selected users' potential ratings to be diverse, so that: (1) selected users tend to have different interest. Ratings of these users would provide more information compared to ratings of similar users, and (2) the final prediction model trained on these users' feedback would not be biased to a fixed region of ratings. To achieve this, we first train a regression model based on $\mathbf{R}$ and $\mathbf{T}$, as shown in Fig. 3c. Features contain users and attributes (without items), and labels are users' ratings. Once the regression model is learned, all users' potential ratings on the new item $P_r(u_m, i_{new}), u_m \in U$ can be estimated. Then the diverse matrix $\mathbf{D} \in R^{|U|\times|U|}$ is defined as

$$D(m,n) = |P_r(u_m, i_{new}) - P_r(u_n, i_{new})|^{\frac{1}{2}}. \tag{3}$$

$D(m,n)$ denotes the diverse value between u_m and u_n. Calculating $\mathbf{D}$ is computationally expensive. However, the calculations of different diverse values are independent with each other. Therefore, when applied to real-world recommender systems, they can be performed parallelly with acceleration techniques such as GPU acceleration [46], distributed computing [47], etc. We tend to select u_m and u_n together if $D(m,n)$ is large.

Criterion (3): Selected Users' Ratings are Objective. A rating is *objective* means that this rating approximates the item's average rating, which is a good estimation of this item's quality [48], [49], [50]. We favor selecting users who always generate objective ratings in the past, then they are expected to also generate objective ratings on i_{new}. We define a vector $\mathbf{o} \in R^{|U|\times 1}$ as

$$o(m) = \frac{1}{\log|I(u_m)|+1} \cdot \frac{1}{|I(u_m)|} \cdot \sum_{i_n \in I(u_m)} (R(m,n) - \overline{R(n)})^2, \tag{4}$$

where $I(u_m)$ is the item set that u_m has rated. $R(m,n)$ is u_m's rating on i_n. $\overline{R(n)}$ is the mean rating on i_n. $\frac{1}{\log|I(u_m)|+1}$ is a penalty for users who have rated few items, since a user may generate a rating that approximates $\overline{R(n)}$ by coincidence. A smaller $o(m)$ indicates that u_m is more objective, so we tend to select u_m if $o(m)$ is small.

With Criterion (3), selected users tend to give higher/lower ratings for items with better/worse quality. Therefore, the re-trained prediction model would generate overall higher/lower prediction ratings for new items with better/worse quality, which is more reasonable. Criterion (3) is a complement to Criterion (2). Criterion (2) encourages the feedback ratings to have a large variance, thus it could increase the model's differentiation power in terms of different users. With Criterion (3), the overall feedback ratings would be higher/lower for items with better/worse quality, which could increase the model's differentiation power in terms of different new items.

Criterion (4): Selected Users are Representative. Here, a selected user is *representative* means that this user is similar to unselected users. We prefer to select representative users so that from their feedback, we can learn more about the preference of unselected users. To achieve this, a similarity matrix $\mathbf{S} \in R^{|U|\times|U|}$ is first defined as

$$S(m,n) = \begin{cases} Sim(R(m,:), R(n,:)) & \text{if } m \neq n \\ 0 & \text{if } m = n \end{cases}, \tag{5}$$

where $R(m,:)$ and $R(n,:)$ are the row vectors of $\mathbf{R}$, representing users u_m and u_n. $Sim(R(m,:), R(n,:))$ is their similarity. We use the cosine similarity in this paper. Acceleration techniques described in Criterion (2) can also be applied to calculating $\mathbf{S}$. We tend to select one of u_m and u_n if $S(m,n)$ is large.

It is worth noting that Criterion (2) and Criterion (4) are highly related to the *avoiding redundancy* and *ensuring representativeness* described in [11], which can be interpreted from the perspective of minimizing the distribution difference between the selected and unselected instances.

We now formulate the user selection task as an explicit mathematical optimization problem, where the objective is to select a batch of users based on above criteria. Specifically, we define a binary vector $\mathbf{q} \in \{0,1\}^{|U|\times 1}$, where each entry $q(m)$ denotes whether u_m will be included in the batch ($q(m) = 1$) or not ($q(m) = 0$). Thus our user selection strategy (with given batch size k) can be expressed as the following integer quadratic programming (IQP) problem

$$\begin{aligned} \max_{\mathbf{q}} \ & \alpha \sum_{m=1}^{|U|} q(m)p(m) + \beta \sum_{m=1}^{|U|}\sum_{n=1}^{|U|} q(m)q(n)D(m,n) \\ & - \gamma \sum_{m=1}^{|U|} q(m)o(m) + \sigma \sum_{m=1}^{|U|}\sum_{n=1}^{|U|} q(m)(1-q(n))S(m,n) \\ & \text{s.t. } q(m) \in \{0,1\}, \forall m \text{ and } \sum_{m=1}^{|U|} q(m) = k. \end{aligned} \tag{6}$$

The first term is to satisfy Criterion (1). Supposing u_m is with high possibility to rate i_{new} (i.e., $p(m)$ is large), then in order to optimize the objective function, u_m is encouraged to be selected (i.e., $q(m)$ is encouraged to be 1). The second term is to satisfy Criterion (2). Supposing potential ratings of u_m and u_n are very diverse (i.e., $D(m,n)$ is large), then u_m and u_n are encouraged to be selected together (i.e., $q(m)$ and $q(n)$ are encouraged to be 1). The third term is to satisfy Criterion (3), with the analysis similar to the first term. Note that we minus this term, since we want to select u_m when $o(m)$ is small. The last term enforces selected users to be similar to unselected users, satisfying Criterion (4), with the analysis similar to the second term. Note that $(1-q(n)) = 1$ indicates that u_n is not selected. α, β, γ and σ are trade-off parameters.

Eq. (6) can be reformulated as

$$\begin{aligned} & \alpha\mathbf{q}^T\mathbf{p} + \beta\mathbf{q}^T\mathbf{D}\mathbf{q} - \gamma\mathbf{q}^T\mathbf{o} + \sigma\mathbf{q}^T\mathbf{S}(\mathbf{1}-\mathbf{q}) \\ & = \alpha\mathbf{q}^T\mathbf{p} + \beta\mathbf{q}^T\mathbf{D}\mathbf{q} - \gamma\mathbf{q}^T\mathbf{o} + \sigma\mathbf{q}^T\mathbf{S}\mathbf{1} - \sigma\mathbf{q}^T\mathbf{S}\mathbf{q} \\ & = \mathbf{q}^T(\alpha\mathbf{p} - \gamma\mathbf{o} + \sigma\mathbf{S}\mathbf{1}) + \mathbf{q}^T(\beta\mathbf{D} - \sigma\mathbf{S})\mathbf{q} \\ & = \mathbf{q}^T diag(\alpha\mathbf{p} - \gamma\mathbf{o} + \sigma\mathbf{S}\mathbf{1})\mathbf{q} + \mathbf{q}^T(\beta\mathbf{D} - \sigma\mathbf{S})\mathbf{q} \\ & = \mathbf{q}^T\mathbf{M}\mathbf{q}, \end{aligned} \tag{7}$$

where $\mathbf{1}$ is a vector with all entries equal to 1 and $diag\,(\alpha\mathbf{p} - \gamma\mathbf{o} + \sigma\mathbf{S1})$ is a diagonal matrix, whose (i,i)th entry is equal to the ith entry of $(\alpha\mathbf{p} - \gamma\mathbf{o} + \sigma\mathbf{S1})$. Since we have the constraint $q(m) \in \{0,1\}, \forall m$, thus we derive $\mathbf{q}^T(\alpha\mathbf{p} - \gamma\mathbf{o} + \sigma\mathbf{S1}) = \mathbf{q}^T diag(\alpha\mathbf{p} - \gamma\mathbf{o} + \sigma\mathbf{S1})\mathbf{q}$. $\mathbf{M}$ is defined as

$$M(m,n) = \begin{cases} \beta D(m,n) - \sigma S(m,n) & \text{if } m \neq n \\ \alpha p(m) - \gamma o(m) + \sigma \mathbf{S1}(m) & \text{if } m = n \end{cases}. \quad (8)$$

Finally, the objective function is transformed to

$$\begin{aligned} &\max_{\mathbf{q}} \mathbf{q}^T \mathbf{M} \mathbf{q}, \\ &\text{s.t. } q(m) \in \{0,1\}, \forall m \text{ and } \sum_{m=1}^{|U|} q(m) = k. \end{aligned} \quad (9)$$

Directly solving this integer quadratic programming (IQP) problem is NP-hard. However, it can be relaxed to (1) a convex quadratic programming (QP) problem by relaxing the constraint $q(m) \in \{0,1\}$ to $q(m) \in [0,1]$ [11], or (2) a convex linear programming (LP) problem of two types, one from [11] and the other from [12]. Here we briefly describe the convex LP solution from [12]. It has two steps. In step 1, compute a vector $\mathbf{v} \in R^{|U|\times 1}$ containing column sums of $\mathbf{M}$ and identify the k largest entries in $\mathbf{v}$ to derive the initial solution $\mathbf{q}_0$ (replace the k largest entries of $\mathbf{v}$ with value 1 and replace the other entries with value 0, then assign it to $\mathbf{q}_0$). In step 2, it is a iterative process as shown in Algorithm 1. Starting with the initial solution $\mathbf{q}_0$, we generate a sequence of solutions $\mathbf{q}_1, \mathbf{q}_2, \ldots$ until convergence. Finally, we get the solution $\mathbf{q}$, whose 1-value entries indicate the selected user set to rate i_{new}.

Algorithm 1. Iterative Process for the LP Solution

1: t = 0
2: **repeat**
3: t = t + 1
4: Compute $\mathbf{q}'_t = \mathbf{M} \cdot \mathbf{q}_{t-1}$
5: Replace the k largest entries of $\mathbf{q}'_t$ with value 1 and replace the other entries with value 0
6: $\mathbf{q}_t = \mathbf{q}'_t$
7: **until** Convergence ($\mathbf{q}_t$ is equal to $\mathbf{q}_{t-1}$)

If $\mathbf{M}$ is positive semi-definite, Algorithm 1 has a guaranteed monotonic convergence. If $\mathbf{M}$ is not positive semi-definite, with a positive scalar added to the diagonal elements, this algorithm can still be run on the shifted quadratic function to guarantee a monotonic convergence [51]. The iterative process converges fast, thus there is only a marginal increase of the running time. Therefore, the complexity of our algorithm is $O(|U|^2)$, where $|U|$ is the number of users. Refer to [12] for more details about the complexity analysis. In real-world recommender systems, $\mathbf{M}$ may be too large for the memory to load. Our algorithm can still work well in this situation. For step 1, the memory only needs to load one column of $\mathbf{M}$ at a time to calculate column sums of $\mathbf{M}$. For each iteration of step 2, it only needs to load one row (equal to the corresponding column since $\mathbf{M}$ is symmetric) of $\mathbf{M}$ and $\mathbf{q}_{t-1}$ at a time to calculate $\mathbf{M} \cdot \mathbf{q}_{t-1}$.

4.2 Dynamic Active Learning Budget

We propose a dynamic active learning budget so that the limited active learning resources can be properly distributed. The total budget is denoted as k_{total}, representing the total number of rating requests. We use $new\_item_1, new\_item_2, \ldots, new\_item_l$ to represent l new items. Their distributed budgets are denoted as $k(1), k(2), \ldots, k(l)$. Thus we have $k_{total} = \sum_{i=1}^{l} k(i)$. We propose to distribute more budgets to new items with the following two features.

First, these items are *popular*. Here, a new item is popular means many people would be willing to rate it. In the active learning phase, we will get more feedback ratings from popular items rather than unpopular items. In the prediction phase, popular items tend to have more test cases. Hence, learning more about popular items, rather than unpopular ones, will improve the overall prediction performance. We use the mean of all users' willing scores to measure it

$$\begin{aligned} &popular(new\_item_i) \\ &= \frac{1}{|U|} \sum_{u_m \in U} willing\_score(u_m, new\_item_i), \\ &i \in \{1,2,\ldots,l\}, \end{aligned} \quad (10)$$

where $willing\_score(u_m, new\_item_i)$ is defined in Section 4.1.

Second, these items are *controversial*. Here, a new item is controversial means we are uncertain about whether they will be liked or disliked by users. For items which will be obviously favored by almost all users, we already have a high confidence what ratings users would give to them. In contrary, it is the controversial items that we need to learn more about. We use the standard deviation of potential ratings to measure it

$$\begin{aligned} &controversial(new\_item_i) \\ &= \frac{1}{|U|} \sqrt{\sum_{u_m \in U} (P_r(u_m, new\_item_i) - \overline{P_r(new\_item_i)})^2}, \\ &i \in \{1,2,\ldots,l\}, \end{aligned} \quad (11)$$

where $P_r(u_m, new\_item_i)$ is defined in Section 4.1. $\overline{P_r(new\_item_i)}$ is the average of potential ratings on new_item_i. A budget score for each new item is defined as

$$\begin{aligned} &budget\_score(new\_item_i) = \\ &popular(new\_item_i) + \lambda \cdot controversial(new\_item_i), \end{aligned} \quad (12)$$

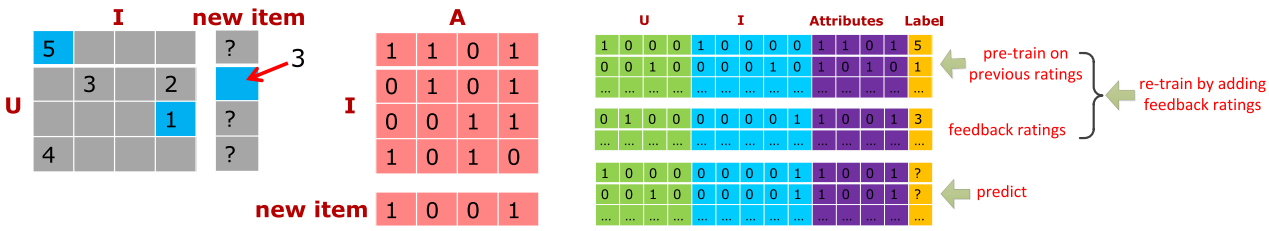
where λ is a parameter to balance the importance of two features. Finally, the budget is distributed as

$$k(i) = \frac{budget\_score(new\_item_i)}{\sum_{j=1}^{l} budget\_score(new\_item_j)} \cdot k_{total}, i \in \{1,2,\ldots,l\}. \quad (13)$$

$k(i)$ are rounded to be integers. Eq. (13) ensures that more popular and controversial items will get more budget, and meanwhile each item has the opportunity to gain some budget.

4.3 Rating Prediction Based on Feedback

Once selected users' feedback is obtained, we use another regression model to predict unselected users' ratings.



(a) A feedback rating is obtained for the new item (b) Factorization Machines for the final prediction

Fig. 4. We first pre-train the regression model using previous ratings. Once feedback ratings for the new item are obtained, we re-train the model to strengthen it. (a) Is to show that a feedback rating from the second user (rating '3') is obtained. (b) Is to show how we re-train the model by adding users' feedback ratings. Finally, ratings of all unselected users are predicted.

Features in this model contain not only users and items' attributes, but also items. The instances contain both previous ratings and the newly obtained feedback ratings. To accelerate the convergence speed, we first pre-train the regression model using previous ratings to get pre-trained parameters. When feedback ratings are obtained, we then use these pre-trained parameters as initial parameters, all previous ratings and feedback ratings as training data, to re-train the model. Finally, ratings of all unselected users are predicted. Fig. 4 shows the detailed procedure.

The procedure of first pre-training and then re-training is similar to the idea of Bayesian Analysis [52]. That is, by pre-training on previous items, we learn users' preferences on attributes, and gain a "prior" understanding of users' preferences on i_{new} based on i_{new}'s attributes. Then users' feedback on i_{new} enhance our understanding of i_{new} and allow us to give "posterior" estimations for users' preferences on i_{new}.

4.4 Exploitation-Exploration Analysis

As described in the introduction section, there are two goals in our task, i.e., *exploitation* (exploiting "existing knowledge" to select users who are willing to rate new items) and *exploration* (selecting users whose feedback can provide as much "new knowledge" about unselected users' preferences as possible and then generate accurate rating predictions for them). (1) The dynamic budget strategy distributes more budget to popular items, which users are more willing to rate. For a certain item, Criterion (1) encourages more willing users to be selected. Thus they both contribute to *exploitation*. (2) The dynamic budget strategy and four criteria all help to learn more about unselected users' preferences and generate more accurate rating predictions. Thus they all contribute to *exploration*. Therefore, our method considers both of *exploitation* and *exploration*. In addition, we are able to adjust the parameter setting to further balance their trade-off. Specifically, once the best prediction accuracy is obtained with all parameters assigned to appropriate values, if we want to attach more importance to *exploitation*, we just need to simply increase α (the weight of Criterion (1)). The reason is that putting more weight on Criterion (1) would result in a higher rate of feedback ratings. However, increasing α will destroy the optimized parameter setting for rating prediction, thus the prediction accuracy would decrease.

TABLE 1
The Movielens-IMDB and Amazon Datasets

	Movielens-IMDB	Amazon
Number of users	5000	973
Number of items	9998	5000
Number of ratings	5154925	97967
Number of attributes	255942	3840

5 Experiments

5.1 Dataset

Our proposed algorithm is evaluated on two datasets, Movielens-IMDB and Amazon. For the Movielens-IMDB dataset, ratings are collected from *Movielens*[3] and attributes of movies are collected from *imdbpy*.[4] *Ratings*[5] and *attributes*[6] are also collected for the Amazon dataset. The statistics of these two datasets are shown in Table 1. In Movielens-IMDB, the number of attributes is the total number of directors, actors, genres, etc. In Amazon, the number of attributes is the total number of authors, publishers, etc. We collect ratings from the Movielens dataset rather than the *official Netflix dataset*.[7] The reason is that items' detailed attributes are required in our setting. The attributes of movies in Movielens can be collected from *http://imdbpy.sourceforge.net/* by accurately linking the movie ids in Movielens and IMDB. However, we did not figure out how to accurately obtain the attributes of movies in Netflix. For each dataset, 20 percent items are randomly chosen as "new items" (i.e., testing items). Ratings and attributes of the other 80 percent items are used to train different models. Our goal is to generate accurate rating predictions on the testing items. Following [30], we randomly select half of all users as the active-selection set and the remaining users form the prediction set. For all testing items, users are selected from the active-selection set for rating requests. The rating prediction and the evaluation are performed in the prediction set.

5.2 Compared Algorithms

Since in this paper, we want to handle new items with no rating, thus many previous active learning recommendation methods [25], [26], [53], which require at least a small amount of initial ratings, are not applicable in our task. [8], [9] are the most related works to ours, which also address the item cold-start problem in an active learning scheme.

3. http://grouplens.org/datasets/movielens/
4. http://imdbpy.sourceforge.net/
5. http://jmcauley.ucsd.edu/data/amazon/links.html
6. https://developer.amazon.com/
7. https://www.kaggle.com/netflix-inc/netflix-prize-data

However, the approach proposed by [8] is under an online setting in which users arrive and are decided to be given rating requests one by one and apparently it is not applicable for our task. [9] assumes that selected users will always rate the new item (the rate of feedback ratings is 100 percent), while our task is under a more realistic setting that only a subset of selected users will give feedback ratings. Thus it is unfair to compare the method in [9] with our method and other baselines. We denote our method without dynamic budget as Factorization Machines with Four Criteria (FMFC) and our method with dynamic budget as FMFC-DB. We try our best to adapt following baselines from previous literature to compare with our proposed methods. HBRNN, LCE and FM are hybrid methods which combine both content and collaborative information. The remaining algorithms all exploit active learning to perform recommendations. The pre-train schedule in Fig. 4 is the same for all active learning methods, but the re-train schedule differs since different active learning methods have different feedback ratings.

Hybrid-Based Recommendation with Nearest Neighbor (HBRNN). This method [54] is a combination of content-based recommendation and item-based collaborative filtering. The similarity between two items i_m, i_n is defined as follows:

$$sim(i_m, i_n) = cos(T(m,:), T(n,:)), \tag{14}$$

where $T(m,:)$ and $T(n,:)$ are row vectors of the item-attribute matrix $\mathbf{T}$, representing i_m and i_n. Then all users' ratings on i_{new} are predicted using an item-based collaborative filtering idea

$$Rating(u_m, i_{new}) = \frac{\sum_{i_n \in I(u_m)} R(u_m, i_n) sim(i_n, i_{new})}{\sum_{i_n \in I(u_m)} sim(i_n, i_{new})}, \quad u_m \in U, \tag{15}$$

where $I(u_m)$ is the item set that u_m rates. $R(u_m, i_n)$ is the rating that u_m gives to i_n.

Local Collective Embeddings (LCE). This method [20] also utilizes the content and collaborative information for recommendation. Different from HBRNN, which is a hybrid-based recommendation method from the nearest neighbor perspective. LCE is a hybrid-based recommendation method from the perspective of matrix factorization. In addition, it exploits the manifold structure of the data to improve the performance. We use the publicly available Matlab implementation[8] of the LCE algorithm. Parameters are set and tuned as recommended in [20].

Factorization Machines Without Active Learning Phase (FM). This method uses Factorization Machines [43] to model user behaviours and items' attributes. FM is equivalent to the pre-trained model in Fig. 4.

Factorization Machines with Random Sampling in the Active Learning Phase (FMRSAL). In this baseline, for i_{new}, k users are randomly selected from the active-selection set for rating requests. Since these users are randomly selected and ratings are sparse in our dataset, thus the rate of feedback ratings is expected to be low. The performance improvement may be limited when compared to FM. However, rating requests are given to users without bias to any type of users, thus no one is always selected for rating requests in this strategy.

8. https://github.com/msaveski/LCE

Factorization Machines with ϵ-Greedy in the Active Learning Phase (FMϵGAL). The ϵ-Greedy algorithm is one solution to the multi-armed bandit problem [35]. We adapt it to our task as follows. For i_{new}, we select k users by k sequential actions. In each action, we select the user who has the highest possibility to rate i_{new} (i.e., u_m with the largest $p(m)$) with probability $1-\epsilon$, and otherwise randomly select other users. The user selected in one action does not participate in following actions. In this strategy, one more parameter, i.e., ϵ, needs to be tuned. When we set $\epsilon = 0$, it is equal to FMFC with only Criterion (1). When we set $\epsilon = 1$, it is transformed into FMRSAL. When we set $0 < \epsilon < 1$, due to the randomness, rating requests are distributed to a wide range of users. Meanwhile, it can ensure a rate of feedback ratings higher than FMRSAL. In our experiments, we find no matter how ϵ varies, the performance in terms of all metrics is always between the performance of FMFC with only Criterion (1) and FMRSAL, thus we only show the experimental results with ϵ equal to 0.5 for simplicity.

Factorization Machines with Poplar Sampling in the Active Learning Phase (FMPSAL). Inspired by [7], [28], for i_{new}, k users who have given the most ratings to the training items are selected for rating requests. Since these users are "frequently" rating users, they also tend to rate i_{new}, which can ensure a high rate of feedback ratings. Note that different from our Criterion (1), which is "personalized" for different new items, users selected in this strategy are always the same.

Factorization Machines with Coverage Sampling in the Active Learning Phase (FMCSAL). Inspired by [24], [28], for i_{new}, k users who have highly co-rated items with other users are selected for rating requests. We define $Coverage(u_i) = \sum_j n_{ij}$, where n_{ij} is the number of items that are rated by both users u_i and u_j. Users with high Coverage values are then selected. The insight is that users co-rate the same items with many other users can better reflect other users' interest, and thus their rating behaviors are more helpful for predicting rating behaviors of other users.

Factorization Machines with Exploration Sampling in the Active Learning Phase (FMESAL). As described in [4], exploration is important for recommendation, especially for new items as in our task. Inspired by [11], [13], for i_{new}, k users are selected for rating requests ensuring that selected users are representative of unselected users, and at the same time, selected users themselves are with high diversity. This can be achieved by optimizing the following objective function:

$$\begin{aligned} &\max_{\mathbf{q}} -\mathbf{q}^T\mathbf{S}\mathbf{q} + \gamma\mathbf{q}^T\mathbf{S}(\mathbf{1}-\mathbf{q}), \\ &\text{s.t. } q(i) \in \{0,1\}, \forall i \text{ and } \sum_{i=1}^{|U|} q(i) = k, \end{aligned} \tag{16}$$

where $\mathbf{q}$ and $\mathbf{S}$ are defined as in Eq. (7). The first term is to ensure "diversity" (selected users are dissimilar to each other) and the second term is for "representative" (selected users are similar to unselected users). This integer quadratic programming (IQP) problem can be relaxed to a standard quadratic problem (QP) and be solved by applying many existing solvers.

5.3 Evaluations

In the active learning phase, we use the following two metrics to measure the user experience of selected users.

Percentage of Feedback Ratings (PFR). The ratio of feedback ratings among all rating requests. It is formally defined as

$$PFR = \frac{\text{Total number of feedback ratings}}{\text{Total number of rating requests}}. \quad (17)$$

Compared to users with no feedback, users giving feedback ratings are more likely to be willing to rate the item. Thus a higher *PFR* indicates a better user experience.

Average Selecting Times (AST). Average selecting times per user after a selection strategy is applied for all testing items. It is formally defined as

$$AST = \frac{\text{Total number of rating requests}}{\text{Total number of distinct users for rating requests}}. \quad (18)$$

A higher AST means some users are always selected for rating requests, which will quickly annoy them and indicates a poorer user experience. For algorithms without active learning, i.e., HBRNN, LCE and FM, these two metrics are not measured.

In the prediction phase, we use *Root Mean Square Error (**RMSE**)* and *Mean Absolute Error (**MAE**)* to measure the user experience of unselected users. They are defined as follows:

$$RMSE = \sqrt{\frac{1}{|R|}\sum_{(u_m,i_{new})\in R}(R(u_m,i_{new}) - \tilde{R}(u_m,i_{new}))^2}, \quad (19)$$

$$MAE = \frac{1}{|R|}\sum_{(u_m,i_{new})\in R}|R(u_m,i_{new}) - \tilde{R}(u_m,i_{new})|, \quad (20)$$

R is the set of (user, item) pairs. $R(u_m, i_{new})$ is the actual rating that u_m gives to i_{new} and $\tilde{R}(u_m, i_{new})$ is the predicted rating.

For methods with no active learning, i.e., HBRNN, LCE and FM, models are directly trained on training items. For the remaining methods, given i_{new}, we select users from the active-selection set to see whether they have actual ratings on i_{new}. If yes, we regard these actual ratings as feedback ratings. Then we exploit all feedback ratings to re-train the model. For all methods, $RMSE$ and MAE are evaluated in the prediction set.

Apart from rating prediction, we can mimic a setting of top-N recommendations as follows. First, for all testing items, we predict ratings in the prediction set. Second, for each user, N (we set $N = 10$) testing items with the largest predicted ratings form the recommendation list. We regard new items with actual ratings larger than 3 as users' preferred items [55]. The following ranking metrics are used to evaluate the performance of top-N recommendations.

Precision, Recall. $Precision$ is defined as the number of correctly recommended items (i.e., the number of preferred items existing in the recommendation list) divided by the number of all recommended items. $Recall$ is defined as the number of correctly recommended items divided by the total number of items which should be recommended (i.e., the number of preferred items). $Precision@k$ and $Recall@k$ are the corresponding values at ranking position k. In our setting, there are $N = 10$ items in the recommendation list, while the number of preferred items is relatively large, thus the original $Recall$ is too small. We multiply it by an appropriate factor to get the modified $Recall$ [56].

*Normalized Discount Cumulative Gain (**NDCG**)*. $NDCG$ at position k is defined as

$$NDCG@k = \frac{1}{IDCG}\times\sum_{i=1}^{k}\frac{2^{r_i}-1}{\log_2(i+1)}, \quad (21)$$

where r_i is the relevance rating of the item at position i. $IDCG$ is set so that the perfect ranking has a $NDCG$ value of 1. In our case, r_i is set to be the actual rating for preferred items and 0 for the other items.

5.4 Parameter Setting

Before setting the parameters, we calibrate the four criteria first. The calibration contains the following two steps.

Step 1: we normalize $\mathbf{p}$, $\mathbf{D}$, $\mathbf{o}$, $\mathbf{S}$ to be $\mathbf{p}'$, $\mathbf{D}'$, $\mathbf{o}'$, $\mathbf{S}'$ by standardization. Specifically, the normalization formula is: $p'_i = \frac{p_i - \bar{\mathbf{p}}}{\sigma_{\mathbf{p}}}$, $D'_{ij} = \frac{D_{ij} - \bar{\mathbf{D}}}{\sigma_{\mathbf{D}}}$, $o'_i = \frac{o_i - \bar{\mathbf{o}}}{\sigma_{\mathbf{o}}}$, $S'_{ij} = \frac{S_{ij} - \mathbf{S}}{\sigma_{\mathbf{S}}}$, where p'_i and p_i are the ith entries of $\mathbf{p}'$ and $\mathbf{p}$, respectively. $\bar{\mathbf{p}}$ and $\sigma_{\mathbf{p}}$ are the mean and standard deviation of all entries in $\mathbf{p}$. D'_{ij} and D_{ij} are entries with the ith row and jth column in $\mathbf{D}'$ and $\mathbf{D}$, respectively. $\bar{\mathbf{D}}$ and $\sigma_{\mathbf{D}}$ are the mean and standard deviation of all entries in $\mathbf{D}$. The denotations for $\mathbf{o}$ and $\mathbf{S}$ are similar to those in $\mathbf{p}$ and $\mathbf{D}$, respectively.

Step 2: we divide $\mathbf{D}'$ and $\mathbf{S}'$ by $|U|$ to be $\mathbf{D}_{new}$ and $\mathbf{S}_{new}$. There are $|U|$, $|U|*|U|$, $|U|$ and $|U|*|U|$ entries in $\mathbf{p}' \in R^{|U|\times 1}$, $\mathbf{D}' \in R^{|U|\times|U|}$, $\mathbf{o}' \in R^{|U|\times 1}$ and $\mathbf{S}' \in R^{|U|\times|U|}$, respectively. If we do not calibrate $\mathbf{p}'$, $\mathbf{D}'$, $\mathbf{o}'$ and $\mathbf{S}'$, then Eq. (6) will be vastly influenced by the second and fourth terms (due to much more entries in them). To prevent $\mathbf{D}'$ and $\mathbf{S}'$ from dominating $\mathbf{p}'$ and $\mathbf{o}'$ when optimizing Eq. (6), we divide $\mathbf{D}'$ and $\mathbf{S}'$ by $|U|$. α, β, γ and σ in Eq. (6) are tuned based on $\mathbf{p}'$, $\mathbf{D}_{new}$, $\mathbf{o}'$, $\mathbf{S}_{new}$.

k, α, β, γ and σ are the main parameters in our paper. k is the number of selected users for active learning. We empirically set $k = 25$ for each testing item to tune the other parameters. There are four parameters α, β, γ and σ to trade off the importance of different criteria. We fix $\alpha = 1$ and tune the other three free parameters by grid search. We use $RMSE$ as the tuning metric, where $RMSE$ is measured by cross-validation on the training data (users are also split to the active-selection set and the prediction set). For the Movielens-IMDB dataset, we finally get $\alpha = 1, \beta = 0.3, \gamma = 0.1, \sigma = 0.1$. We regard the performance measured in this parameter setting as the performance of FMFC. Furthermore, we fix $\alpha = 1, \beta = 0.3, \gamma = 0.1, \sigma = 0.1$, and implement FMFC-DB, i.e., our method with dynamic active learning budget. The total budget k_{total} (see Section 4.2) is set to be $25 \times l$, where l is the number of testing items. For the Amazon dataset, we get $\alpha = 1, \beta = 0.3, \gamma = 0.08, \sigma = 0.2$. The other settings are the same as for the Movielens-IMDB dataset.

5.5 Results and Analysis

5.5.1 Algorithm Comparison

For each dataset, we repeat the training-testing experiments for 100 times (the testing item set is independently sampled

TABLE 2
Performance Comparison of Active Learning and Rating Prediction on Movielens-IMDB

	$PFR(\%)$	AST	$RMSE$	MAE
HBRNN	x	x	0.8792	0.6738
LCE	x	x	0.8754	0.6712
FM	x	x	1.0364	0.7828
FMRSAL	5.17	**19.98**	0.9244	0.7320
FMϵGAL($\epsilon = 0.5$)	14.24	23.62	0.8728	0.6701
FMPSAL	20.32	1998	0.8520	0.6562
FMCSAL	21.66	1998	0.8511	0.6549
FMESAL	6.35	1998	0.9149	0.7043
FMFC	23.04	163.76	0.8305	0.6388
FMFC-DB	**23.78**	140.03	**0.8231**	**0.6308**

TABLE 3
Performance Comparison of Active Learning and Rating Prediction on Amazon

	$PFR(\%)$	AST	$RMSE$	MAE
HBRNN	x	x	0.8496	0.6565
LCE	x	x	0.8489	0.6564
FM	x	x	0.8781	0.6709
FMRSAL	2.09	**51.33**	0.8760	0.6619
FMϵGAL($\epsilon = 0.5$)	8.04	55.87	0.8489	0.6561
FMPSAL	8.56	1000	0.8449	0.6533
FMCSAL	8.61	1000	0.8418	0.6511
FMESAL	7.63	1000	0.8540	0.6582
FMFC	13.82	82.68	0.8195	0.6342
FMFC-DB	**14.84**	71.31	**0.8052**	**0.6289**

each time and the parameters in our methods are set to be the same as in Section 5.4) and the averaged performance is reported. As shown in Tables 2 and 3, our methods (FMFC and FMFC-DB) outperform the other baselines in terms of $RMSE$ and MAE, which indicates our methods have the highest prediction accuracy. Our methods also perform the best in the task of top-N recommendations according to Tables 4 and 5. FM with different active learning strategies perform better than FM. This is because feedback ratings give us more understanding of testing items, which can enhance the prediction model. As methods without active learning, HBRNN and LCE perform better than FM and even better than active learning methods FMRSAL and FMESAL. The reason may be that HBRNN and LCE can make better use of both content and collaborative information than FM. LCE has a slightly better performance than HBRNN. The reason may be that it exploits the manifold structure to improve the recommendation performance. FMPSAL, FMCSAL, FMFC and FMFC-DB perform better than the other three active learning methods. The main reason is that these four methods can ensure a high rate of feedback ratings, which is the domain factor that influences the prediction accuracy (analyzed in the next section). Our methods outperform FMPSAL and FMCSAL because (1) our methods achieve higher rates of feedback ratings, and (2) our methods consider not only the rate of feedback ratings (Criterion (1)), but also other three factors (Criteria (2), (3), (4)). There is no active learning phase for HBRNN, LCE and FM, thus we compare the other methods in terms of PFR and AST. FMRSAL and FMESAL have rather low PFR, which indicates they always select unwilling users to rate new items. FMϵGAL has a relatively higher PFR. The

TABLE 4
Performance Comparison of Top-N Recommendations on Movielens-IMDB

	$Precision@5$	$Precision@10$	$Recall@5$	$Recall@10$	$NDCG@5$	$NDCG@10$
HBRNN	0.3554	0.2775	0.1042	0.1791	0.2806	0.3617
LCE	0.3593	0.2804	0.1054	0.1810	0.2799	0.3653
FM	0.2835	0.2151	0.0831	0.1388	0.2062	0.2884
FMRSAL	0.3017	0.2324	0.0885	0.1499	0.2252	0.3087
FMϵGAL($\epsilon = 0.5$)	0.3601	0.2821	0.1055	0.1821	0.2801	0.3679
FMPSAL	0.3787	0.3025	0.1111	0.1952	0.3302	0.4114
FMCSAL	0.3869	0.3086	0.1134	0.1991	0.3298	0.4182
FMESAL	0.3208	0.2519	0.0941	0.1625	0.2497	0.3312
FMFC	0.4486	0.3591	0.1316	0.2317	0.3916	0.4738
FMFC-DB	**0.4791**	**0.3898**	**0.1405**	**0.2515**	**0.4436**	**0.5241**

TABLE 5
Performance Comparison of Top-N Recommendations on Amazon

	$Precision@5$	$Precision@10$	$Recall@5$	$Recall@10$	$NDCG@5$	$NDCG@10$
HBRNN	0.2638	0.2061	0.2162	0.3747	0.2871	0.3732
LCE	0.2671	0.2080	0.2189	0.3782	0.2903	0.3764
FM	0.2004	0.1437	0.1643	0.2613	0.2121	0.2947
FMRSAL	0.2195	0.1603	0.1799	0.2915	0.2324	0.3176
FMϵGAL($\epsilon = 0.5$)	0.2673	0.2101	0.2191	0.3820	0.2953	0.3845
FMPSAL	0.2879	0.2310	0.2360	0.4200	0.3345	0.4201
FMCSAL	0.2933	0.2385	0.2404	0.4336	0.3404	0.4199
FMESAL	0.2388	0.1825	0.1957	0.3318	0.2543	0.3393
FMFC	0.3513	0.2905	0.2858	0.5212	0.4039	0.4823
FMFC-DB	**0.3894**	**0.3273**	**0.3192**	**0.5863**	**0.4521**	**0.5372**

TABLE 6
PFR on Movielens-IMDB and Amazon

	$PFR(\%)$ on Movielens-IMDB		$PFR(\%)$ on Amazon	
	FMFC	FMFC-DB	FMFC	FMFC-DB
No Criterion (1)	9.91	10.16	2.25	2.57
Original	**23.04**	**23.78**	**13.87**	**14.91**

TABLE 7
The Average Diverse Value of Selected Users' Ratings on Movielens-IMDB and Amazon

	The Average Diverse Value on Movielens-IMDB		The Average Diverse Value on Amazon	
	FMFC	FMFC-DB	FMFC	FMFC-DB
No Criterion (2)	1.21	1.23	1.13	1.16
Original	**1.37**	**1.39**	**1.19**	**1.22**

TABLE 8
The Difference between the Average Rating of Selected Users and the Average Rating of All Users on Movielens-IMDB and Amazon

	The Difference on Movielens-IMDB		The Difference on Amazon	
	FMFC	FMFC-DB	FMFC	FMFC-DB
No Criterion (3) ($\lvert\bar{r}_{no\_c3} - \bar{r}_{all}\rvert$)	1.22	1.19	1.43	1.39
Original ($\lvert\bar{r}_{ours} - \bar{r}_{all}\rvert$)	**0.97**	**0.89**	**1.22**	**1.19**

TABLE 9
The Average Similarity between Selected and Unselected Users on Movielens-IMDB and Amazon

	The Average Similarity on Movielens-IMDB		The Average Similarity on Amazon	
	FMFC	FMFC-DB	FMFC	FMFC-DB
No Criterion (4)	0.54	0.56	0.46	0.49
Original	**0.61**	**0.67**	**0.52**	**0.58**

other active learning methods all gain a considerable PFR. For AST, due to the randomness, FMRSAL and FMϵGAL undoubtedly achieve the lowest and second lowest AST. FMPSAL, FMCSAL and FMESAL select the same user set to rate all testing items and will certainly annoy them. Overall, our methods are the best when considering all these metrics. When comparing FMFC with FMFC-DB, we can see that our proposed dynamic active learning budget can further improve the performance in terms of all metrics.

The significant test is performed to show whether the differences of our experimental results are statistically significant. Paired t-tests are conducted to compare the differences between the experiment performance of (1) our two proposed methods (i.e., FMFC and FMFC-DB), (2) FMFC and all baselines, and (3) FMFC-DB and all baselines. Specifically, for each dataset, the training-testing experiments are conducted for 100 times. Then given a certain metric, each method would generate 100 metric values. The inputs of paired t-test are two lists of metric values, with each list corresponding to one compared method. Since we want to validate that one method is better than the other, we use one-tailed hypothesis. The statistical significance is set at $p < 0.05$. The results show that in terms of all metrics except for AST, (1) FMFC-DB has significantly better performance than FMFC, and (2) compared to all baselines, both FMFC and FMFC-DB have significantly better performance.

5.5.2 Criteria Analysis

As mentioned in Section 4.1, to generate more accurate rating predictions of the new item, our methods select users based on four criteria. In this section, we first validate whether each criterion works as claimed. Then we validate their contributions to the final prediction performance.

Criterion (1): Selected users are with high possibility to rate $\mathbf{i_{new}}$

We remove Criterion (1) in FMFC and FMFC-DB to see how PFR varies. As shown in Table 6, without Criterion (1), PFR decreases dramatically for both FMFC and FMFC-DB. Since a higher PFR means that more selected users rate i_{new}, the result validates the effectiveness of Criterion (1).

Criterion (2): Selected users' potential ratings are diverse

The purpose of selecting users with diverse potential ratings is to ensure these users' actual ratings also tend to be diverse. We remove Criterion (2) in FMFC and FMFC-DB to see how the average diverse value (defined in Eq. (3)) of selected users' actual ratings varies. As shown in Table 7, without Criterion (2), the average diverse value decreases for both FMFC and FMFC-DB. The result validates the effectiveness of Criterion (2).

Criterion (3): Selected users' generated ratings are objective

The insight of Criterion (3) is to let the average rating of users selected by our methods (denoted as $\bar{r}_{ours}$) approximate the average rating of all users (denoted as $\bar{r}_{all}$). We measure the difference between $\bar{r}_{ours}$ and $\bar{r}_{all}$, i.e., $|\bar{r}_{ours} - \bar{r}_{all}|$. Meanwhile, we calculate the average rating of users selected without Criterion (3) (denoted as $\bar{r}_{no\_c3}$) and measure $|\bar{r}_{no\_c3} - \bar{r}_{all}|$. As shown in Table 8, we have $|\bar{r}_{no\_c3} - \bar{r}_{all}| > |\bar{r}_{ours} - \bar{r}_{all}|$ for both FMFC and FMFC-DB, which indicates that Criterion (3) can actually make the average rating of selected users closer to $\bar{r}_{all}$.

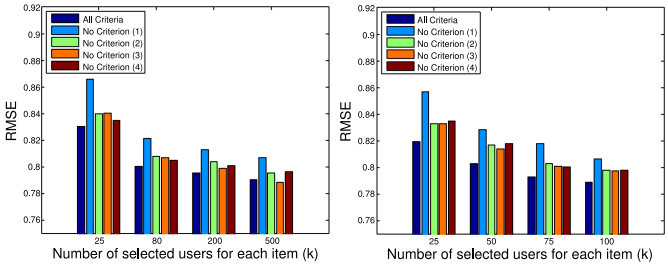
Criterion (4): Selected users are representative

The insight of Criterion (4) is to let selected users similar to unselected users. We measure the average similarity between selected and unselected users with/without Criterion (4). As shown in Table 9, without Criterion (4), the average similarity declines, which indicates that Criterion (4) can actually make selected users more similar to unselected users.

We further validate the contribution of each criterion to the final prediction performance. The results are shown in Fig. 5. $RMSE$ increases when we remove each criterion, which indicates each criterion contributes to the prediction improvement. $RMSE$ increases the most when we remove Criterion (1), which indicates this criterion is a domain factor that influences the prediction improvement. k is the number of selected users. $RMSE$ decreases when k increases. The reason is that larger k leads to more feedback ratings, which will give us more understanding of the new item and thus generate more accurate predictions.

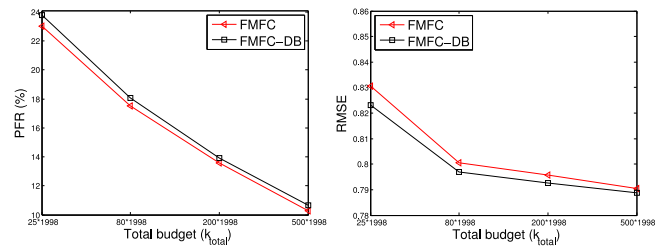
5.5.3 Dynamic Budget Analysis

As mentioned in Section 4.2, we use the strategy of dynamic budget to properly distribute limited active learning

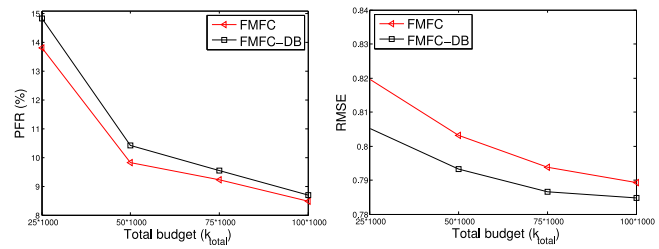


(a) Movielens-IMDB (b) Amazon

Fig. 5. Performance of FMFC measured by $RMSE$ when we remove four criteria one at a time and vary k. *All Criteria* refers to the performance measured when considering all four criteria. *No Criterion (1)* refers to the performance measured when we remove Criterion (1) (i.e., $\alpha = 0$). Similarly, *No Criterion (2)*, *No Criterion (3)*, and *No Criterion (4)* refer to performance measured under the corresponding settings.



(a) PFR on Movielens-IMDB (b) $RMSE$ on Movielens-IMDB



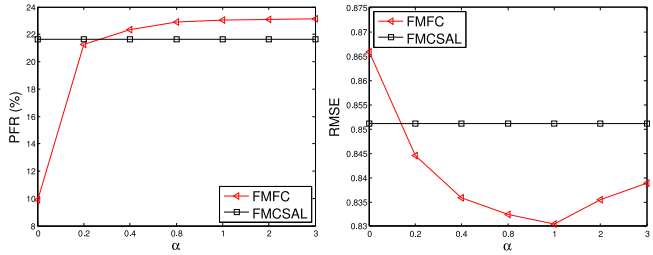
(c) PFR on Amazon (d) $RMSE$ on Amazon

Fig. 6. Performance of FMFC and FMFC-DB measured by PFR (for the active learning phase) and $RMSE$ (for the prediction phase) when we vary the total active learning budget.

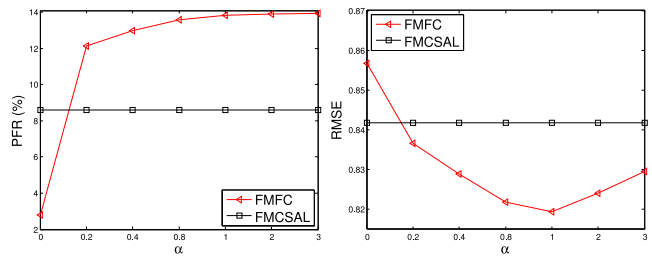
resources. As shown in Fig. 6, for different k_{total} (the total budget), the dynamic budget all contributes to the performance improvement in terms of both PFR and $RMSE$. The improvement is narrowed when k_{total} increases. This is because the dynamic budget is proposed to address the problem of limited active learning resources. When k_{total} is large enough, this strategy provides less help.

5.5.4 Exploitation-Exploration Analysis

Results in Tables 2 and 3 have shown that our method can achieve high performance for both *exploitation* (high PFR in the active learning phase) and *exploration* (low $RMSE$ and MAE in the prediction phase). Now we analyze how α, i.e., the weight of Criterion (1), can further balance their trade-off. We vary α but fix other tuned parameters to see how the performance changes. As shown in Fig. 7, PFR of FMFC keeps increasing when α varies. $RMSE$ first decreases then increases when α varies, and obtains the best result when α is around 1. We certainly will not set $\alpha < 1$, which will achieve poor performance in terms of both $RMSE$ and PFR. For $\alpha \geq 1$, if we attach more importance to the active learning phase, we need to assign a larger value to α.



(a) PFR on Movielens-IMDB (b) $RMSE$ on Movielens-IMDB



(c) PFR on Amazon (d) $RMSE$ on Amazon

Fig. 7. Performance of FMFC and FMCSAL (the best compared method) measured by PFR (for the active learning phase) and $RMSE$ (for the prediction phase) when we vary α.

Similarly, if we pay more attention to the prediction phase, then a smaller value is assigned. These experimental results are consistent with the analysis of Section 4.4.

6 Conclusion

In this paper, we propose a novel recommendation scheme for the item cold-start problem by leveraging both active learning and items' attribute information. We first pre-train the rating prediction model with users' historical ratings and items' attributes. Second, given a new item, a small portion of users are selected to rate this item based on four useful criteria. Third, the prediction model is re-trained by adding feedback ratings. Finally, unselected users' ratings are predicted by the re-trained model. We further propose a dynamic active learning budget to properly distribute active learning resources, which improves the overall recommendation performance. The idea of dynamic active learning budget can also be applied to other active learning related tasks. Our methods can ensure a relatively good user experience for both of selected users in the active learning phase and unselected users in the prediction phase. For future work, we will explore more other criteria to improve our user selection strategy. In addition, the sequential information of user behaviors [57], [58] and the ideas from Multi-task [59] (e.g., co-train the classification and regression tasks), GAN [60] (e.g., refine the negative instances) could also be incorporated in our method. Finally, libFM used in this paper is a regression model, which is suitable for the task of rating prediction. We will try to expend our method with a ranking model to better address the top-N recommendation task.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China (Grant Nos: 61751307, 61522206, 61373118, 61672409, 61876144, 61876145), the National Youth Top-notch Talent Support Program, the Major Basic Research

Project of Shaanxi Province (Grant No. 2017ZDJC-31), the Science and Technology Plan Program in Shaanxi Province of China (Grant No. 2017KJXX-80), and the Shaanxi Province Science Fund for Distinguished Young Scholars (Grant No. 2018JC-016).

REFERENCES

[1] L. Hong, A. S. Doumith, and B. D. Davison, "Co-factorization machines: Modeling user interests and predicting individual decisions in twitter," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2013, pp. 557–566.

[2] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations," in *Proc. IEEE Int. Conf. Data Mining*, 2010, pp. 176–185.

[3] S. Hauger, K. H. Tso, and L. Schmidt-Thieme, "Comparison of recommender system algorithms focusing on the new-item and user-bias problem," in *Data Analysis, Machine Learning and Applications*. Berlin, Germany: Springer, 2008, pp. 525–532.

[4] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan, "Active learning in recommender systems," in *Recommender Systems Handbook*. Berlin, Germany: Springer, 2015, pp. 809–846.

[5] M. Elahi, F. Ricci, and N. Rubens, "Active learning in collaborative filtering recommender systems," in *E-Commerce and Web Technologies*. Berlin, Germany: Springer, 2014, pp. 113–124.

[6] A. M. Rashid, G. Karypis, and J. Riedl, "Learning preferences of new users in recommender systems: An information theoretic approach," *ACM SIGKDD Explorations Newslett.*, vol. 10, no. 2, pp. 90–100, 2008.

[7] N. Golbandi, Y. Koren, and R. Lempel, "Adaptive bootstrapping of recommender systems using decision trees," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2011, pp. 595–604.

[8] M. Aharon, O. Anava, N. Avigdor-Elgrabli, D. Drachsler-Cohen, S. Golan, and O. Somekh, "Excuseme: Asking users to help in item cold-start recommendations," in *Proc. ACM Conf. Recommender Syst.*, 2015, pp. 83–90.

[9] O. Anava, S. Golan, N. Golbandi, Z. Karnin, R. Lempel, O. Rokhlenko, and O. Somekh, "Budget-constrained item cold-start handling in collaborative filtering recommenders via optimal design," in *Proc. Int. Conf. World Wide Web*, 2015, pp. 45–54.

[10] Z. Huang, "Selectively acquiring ratings for product recommendation," in *Proc. Int. Conf. Electron. Commerce*, 2007, pp. 379–388.

[11] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, "Batch mode active sampling based on marginal probability distribution matching," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 741–749.

[12] S. Chakraborty, V. Balasubramanian, A. R. Sankar, S. Panchanathan, and J. Ye, "BatchRank: A novel batch mode active learning framework for hierarchical classification," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 99–108.

[13] N. Rubens and M. Sugiyama, "Influence-based collaborative active learning," in *Proc. ACM Conf. Recommender Syst.*, 2007, pp. 145–148.

[14] L. Rokach, L. Naamani, and A. Shmilovici, "Pessimistic cost-sensitive active learning of decision trees for profit maximizing targeting campaigns," *Data Mining Knowl. Discovery*, vol. 17, no. 2, pp. 283–316, 2008.

[15] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 19–28.

[16] A. Gunawardana and C. Meek, "Tied Boltzmann machines for cold start recommendations," in *Proc. ACM Conf. Recommender Syst.*, 2008, pp. 19–26.

[17] S.-T. Park and W. Chu, "Pairwise preference regression for cold-start recommendation," in *Proc. ACM Conf. Recommender Syst.*, 2009, pp. 21–28.

[18] M. Nasery, M. Braunhofer, and F. Ricci, "Recommendations with optimal combination of feature-based and item-based preferences," in *Proc. Conf. User Model. Adaptation Personalization*, 2016, pp. 269–273.

[19] A. Gunawardana and C. Meek, "A unified approach to building hybrid recommender systems," in *Proc. ACM Conf. Recommender Syst.*, 2009, pp. 117–124.

[20] M. Saveski and A. Mantrach, "Item cold-start recommendations: Learning local collective embeddings," in *Proc. ACM Conf. Recommender Syst.*, 2014, pp. 89–96.

[21] M. Aharon, A. Kagian, R. Lempel, and Y. Koren, "Dynamic personalized recommendation of comment-eliciting stories," in *Proc. ACM Conf. Recommender Syst.*, 2012, pp. 209–212.

[22] N. Aizenberg, Y. Koren, and O. Somekh, "Build your own music recommender by modeling internet radio streams," in *Proc. Int. Conf. World Wide Web*, 2012, pp. 1–10.

[23] M. Elahi, F. Ricci, and N. Rubens, "A survey of active learning in collaborative filtering recommender systems," *Comput. Sci. Rev.*, vol. 20, pp. 29–50, 2016.

[24] N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 1805–1808.

[25] T. Hofmann, "Collaborative filtering via Gaussian probabilistic latent semantic analysis," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2003, pp. 259–266.

[26] R. Jin and L. Si, "A Bayesian approach toward active learning for collaborative filtering," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, 2004, pp. 278–285.

[27] S. A. Danziger, J. Zeng, Y. Wang, R. K. Brachmann, and R. H. Lathrop, "Choosing where to look next in a mutation sequence space: Active learning of informative P53 cancer rescue mutants," *Bioinf.*, vol. 23, no. 13, pp. i104–i114, 2007.

[28] N. Rubens, R. Tomioka, and M. Sugiyama, "Output divergence criterion for active learning in collaborative settings," *IPSJ Online Trans.*, vol. 2, pp. 240–249, 2009.

[29] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 1289–1296.

[30] A. S. Harpale and Y. Yang, "Personalized active learning for collaborative filtering," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2008, pp. 91–98.

[31] C. E. Mello, M.-A. Aufaure, and G. Zimbrao, "Active learning driven by rating impact analysis," in *Proc. ACM Conf. Recommender Syst.*, 2010, pp. 341–344.

[32] M. Elahi, F. Ricci, and N. Rubens, "Adapting to natural rating acquisition with combined active learning strategies," in *Proc. Int. Symp. Methodologies Intell. Syst.*, 2012, pp. 254–263.

[33] B. Wang, M. Ester, J. Bu, Y. Zhu, Z. Guan, and D. Cai, "Which to view: Personalized prioritization for broadcast emails," in *Proc. Int. Conf. World Wide Web*, 2016, pp. 1181–1190.

[34] B. Wang, M. Ester, Y. Liao, J. Bu, Y. Zhu, Z. Guan, and D. Cai, "The million domain challenge: Broadcast email prioritization by cross-domain recommendation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1895–1904.

[35] M. N. Katehakis and A. F. Veinott, "The multi-armed bandit problem: Decomposition and computation," *Math. Operations Res.*, vol. 12, no. 2, pp. 262–268, 1987.

[36] J. Langford and T. Zhang, "The epoch-greedy algorithm for multi-armed bandits with side information," *Ad. Neural Inf. Process. Syst.*, pp. 817–824, 2008.

[37] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *J. Mach. Learn. Res.*, vol. 3, no. Nov., pp. 397–422, 2002.

[38] V. Dani, T. P. Hayes, and S. M. Kakade, "Stochastic linear optimization under bandit feedback," in *Proc. Conf. Learn. Theory*, 2008, pp. 355–366.

[39] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 2312–2320.

[40] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[41] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Proc. Conf. Learn. Theory*, 2012, pp. 39–1.

[42] D. Russo and B. Van Roy, "Learning to optimize via posterior sampling," *Math. Operations Res.*, vol. 39, no. 4, pp. 1221–1243, 2014.

[43] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, 2012, Art. no. 57.

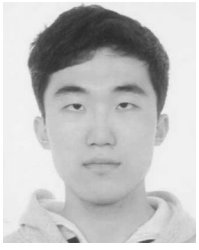
[44] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 635–644.

[45] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 426–434.

[46] D. Tarditi, S. Puri, and J. Oglesby, "Accelerator: Using data parallelism to program GPUs for general-purpose uses," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 5, pp. 325–335, 2006.

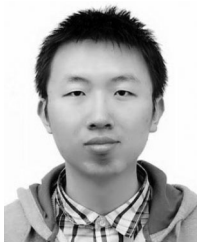
[47] S. H. Bokhari, *Assignment Problems in Parallel and Distributed Computing*. Berlin, Germany: Springer, 2012.
[48] W. Duan, B. Gu, and A. B. Whinston, "Do online reviews matter? An empirical investigation of panel data," *Decision Support Syst.*, vol. 45, no. 4, pp. 1007–1016, 2008.
[49] J. A. Chevalier and D. Mayzlin, "The effect of word of mouth on sales: Online book reviews," *J. Marketing Res.*, vol. 43, no. 3, pp. 345–354, 2006.
[50] C. Dellarocas, N. Awad, and X. Zhang, "Exploring the value of online reviews to organizations: Implications for revenue forecasting and planning," in *Proc. Int. Conf. Inf. Syst.*, 2004, Art. no. 30.
[51] X.-T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *J. Mach. Learn. Res.*, vol. 14, no. Apr., pp. 899–925, 2013.
[52] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*. Berlin, Germany: Springer, 2013.
[53] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *Proc. Int. Conf. Mach. Learn.*, 2000, pp. 839–846.
[54] L. Iaquinta, A. L. Gentile, P. Lops, M. de Gemmis, and G. Semeraro, "A hybrid content-collaborative recommender system integrated into an electronic performance support system," in *Proc. 7th Int. Conf. Hybrid Intell. Syst.*, 2007, pp. 47–52.
[55] Z. Guan, L. Chen, W. Zhao, Y. Zheng, S. Tan, and D. Cai, "Weakly-supervised deep learning for customer review sentiment classification," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 3719–3725.
[56] Y. Zhu, Z. Guan, S. Tan, H. Liu, D. Cai, and X. He, "Heterogeneous hypergraph embedding for document recommendation," *Neurocomput.*, vol. 216, pp. 150–162, 2016.
[57] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by Time-LSTM," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3602–3608.
[58] Y. Zhu, J. Zhu, J. Hou, Y. Li, B. Wang, Z. Guan, and D. Cai, "A brand-level ranking system with the customized attention-GRU model," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3947–3953.
[59] Y. Gong, X. Luo, Y. Zhu, W. Ou, Z. Li, M. Zhu, K. Q. Zhu, L. Duan, and X. Chen, "Deep cascade multi-task learning for slot filling in chinese e-commerce shopping guide assistant," arXiv preprint: arXiv:1803.11326, 2018.
[60] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 515–524.

Yu Zhu received the BS and PhD degrees in computer science from Zhejiang University, China, in 2013 and 2018, respectively. He is currently a senior algorithm engineer with the Alibaba Group. His research interests include machine learning, data mining, and recommender systems.



Jinghao Lin received the BS degree from Zhejiang University, China, in 2015. He is currently working toward the master's degree in the State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, China. His research interests include data mining and recommendation systems.

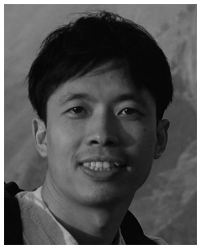
Shibi He is working toward the PhD degree at the University of Illinois at Urbana-Champaign. Before that, he was a member of the Social Network Group in the State Key Lab of CAD&CG under the supervision of professor Deng Cai. His research interests include deep reinforcement learning, game theory, social network, bioinformatics, and computer vision.



Beidou Wang received the BS degree from Zhejiang University, China, in 2011. He is currently in a duo PhD program of Zhejiang University, China and Simon Fraser University, Canada. His research interests include social network mining and recommender systems.

Ziyu Guan received the BS and PhD degrees in computer science from Zhejiang University, China, in 2004 and 2010, respectively. He worked as a research scientist with the University of California at Santa Barbara from 2010 to 2012. He is currently a full professor with the School of Information and Technology of Northwest University, China. His research interests include attributed graph mining and search, machine learning, expertise modeling and retrieval, and recommender systems.

Haifeng Liu received the bachelor's degree in computer science from the Special Class for the Gifted Young from the University of Science and Technology of China, and the PhD degree from the Department of Computer Science, University of Toronto, in 2009. She is an associate professor with the College of Computer Science, Zhejiang University, China. Her research interests lie in the field of machine learning, pattern recognition, and web mining.



Deng Cai received the bachelor's and master's degrees in automation from Tsinghua University, in 2000 and 2003, respectively, and the PhD degree in computer science from the University of Illinois at Urbana Champaign, in 2009. He is a professor with the State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, China. His research interests include machine learning, data mining, and information retrieval. He is a member of the IEEE.