


A Ranking-oriented Collaborative Filtering: A Listwise Approach

Shuaiqiang Wang

Related papers

[Download a PDF Pack](#) of the best related papers 



[Listwise Collaborative Filtering](#)

Shuaiqiang Wang

[VSRank](#)

Shuaiqiang Wang

[VSRank: A Novel Framework for Ranking-Based Collaborative Filtering](#)

Shuaiqiang Wang

Ranking-oriented Collaborative Filtering: A Listwise Approach

SHUAIQIANG WANG, University of Jyväskylä

SHANSHAN HUANG, Shandong University

TIE-YAN LIU, Microsoft Research Asia

JUN MA, Shandong University

ZHUMIN CHEN, Shandong University

JARI VEIJALAINEN, University of Jyväskylä

Collaborative filtering (CF) is one of the most effective techniques in recommender systems, which can be either rating-oriented or ranking-oriented. Ranking-oriented CF algorithms demonstrated significant performance gains in terms of ranking accuracy, being able to estimate a precise preference ranking of items for each user rather than the absolute ratings (as rating-oriented CF algorithms do). Conventional memory-based ranking-oriented CF can be referred to as *pairwise* algorithms. They represent each user as a set of preferences on each pair of items for similarity calculations and predictions. In this study, we propose ListCF, a novel *listwise* CF paradigm that seeks improvement in both accuracy and efficiency in comparison with pairwise CF. In ListCF, each user is represented as a probability distribution of the permutations over rated items based on the Plackett-Luce model, and the similarity between users is measured based on the Kullback–Leibler (KL) divergence between their probability distributions over the set of commonly rated items. Given a target user and the most similar users, ListCF directly predicts a total order of items for each user based on similar users' probability distributions over permutations of the items. Besides, we also reveal insightful connections among pointwise, pairwise and listwise CF algorithms from the perspective of the matrix representations. In addition, to make our algorithm more scalable and adaptive, we present an incremental algorithm for ListCF, which allows incrementally updating the similarities between users when certain user submits a new rating or updates an existing rating. Extensive experiments on benchmark datasets in comparison with the state-of-the-art approaches demonstrate the promise of our approach.

CCS Concepts: • **Information systems** → **Recommender systems**; *Retrieval models and ranking*;

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Collaborative filtering, ranking-oriented collaborative filtering, recommender systems

ACM Reference Format:

Shuaiqiang Wang, Shanshan Huang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen, 2015. Ranking-oriented Collaborative Filtering: A Listwise Approach. *ACM Trans. Inf. Syst.* V, N, Article A (January YYYY), 29 pages.

DOI: 0000001.0000001

1. INTRODUCTION

Ever since the thriving of the Web, the world has been flooded with an overwhelming amount of information. Such a high volume of information has become increasingly

A preliminary version of this paper was published in the Proceedings of the 38st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 343-352, Santiago, Chile, August 9-13, 2015 [Huang et al. 2015b].

Author's addresses: S. Wang and J. Veijalainen, Department of Computer Science and Information Systems, University of Jyväskylä; S. Huang, J. Ma and Z. Chen, School of Computer Science and Technology, Shandong University; T.-Y. Liu, Microsoft Research Asia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© YYYY ACM. 1046-8188/YYYY/01-ARTA \$15.00

DOI: 0000001.0000001

unmanageable and created “a poverty of attention and a need to allocate that attention efficiently” [Simon 1971]. As an effective technique addressing the *information overload* problem, recommender system provides recommendations from a large collection of items based on users’ historical behaviors or preferences. In recent years, recommender system has been widely and successfully applied in many online websites, such as eBay¹, Amazon² and Netflix.

Collaborative filtering (CF) is one of the most successful recommendation techniques to build recommender systems. CF is based on the assumption that users will rate items similarly or have similar behaviors in the future if they rated or acted similarly in the past [Goldberg et al. 1992; Resnick et al. 1994]. Given the success of CF, many algorithms have been proposed, which mainly fall into two categories: rating-oriented CF and ranking-oriented CF.

1.1. Rating-Oriented CF: The Pointwise CF

Rating-oriented CF, such as user-based CF [Breese et al. 1998; Herlocker et al. 2002], predicts a user’s potential ratings on unrated items based on the rating information from other similar users. The similarity between two users is calculated based on their rating scores on the set of commonly rated items. A popular similarity measure is the Pearson correlation coefficient [Herlocker et al. 2002]. Since this technique focuses on predicting each user’s rating on an unrated item, we refer to it as *pointwise* CF. Please note that pointwise CF might not be able to achieve the ultimate goal of recommender systems, which is to present a ranking or recommendation list to a user rather than predict the absolute value of the ratings [Shi et al. 2010]. In fact, as illustrated in the following example, improvements in pointwise error metrics such as RMSE (Root Mean Squared Error) do not necessarily lead to improved ranking effectiveness [Cremonesi et al. 2010].

EXAMPLE 1.1. Suppose $I = \{i_1, i_2, i_3\}$ are the items to recommend to the target user u . The true ratings of u to these items are $\{3, 4, 5\}$. Consider two predictions on I : The first prediction $P_1 = \{1, 2, 3\}$, and the second prediction $P_2 = \{5, 4, 3\}$. Their errors in terms of RMSE are:

$$RMSE_{P_1} = \sqrt{\frac{(3-1)^2 + (4-2)^2 + (5-3)^2}{3}} = 2,$$

$$RMSE_{P_2} = \sqrt{\frac{(3-5)^2 + (4-4)^2 + (5-3)^2}{3}} \approx 1.6$$

Although the error of P_1 is smaller, obviously the ranking performance of P_2 is better, as P_1 predicts a consistent ranking with the ground-truth while P_2 makes a completely opposite ranking to the ground-truth.

1.2. Ranking-Oriented CF: The Pairwise CF

Different from rating-oriented CF, ranking-oriented CF algorithms [McNee et al. 2006] directly generate a preference ordering of items for each user without the intermediate step of rating prediction. Related works include memory-based algorithms such as EigenRank [Liu and Yang 2008] and VSRank [Wang et al. 2012b; 2014], and model-based algorithms such as CoFiRank [Weimer et al. 2007], ListRank-MF [Shi et al. 2010] and CCF [Yang et al. 2011]. In this paper, we focus on memory-based CF algorithms since they have demonstrated many advantages such as strong robustness, interpretability, and competitive performance [Hofmann 2004].

¹<http://www.ebay.com/>

²<http://www.amazon.com/>

Existing memory-based ranking-oriented CF, such as EigenRank [Liu and Yang 2008] and VSRank [Wang et al. 2012b; 2014], can be referred to as *pairwise* ranking-oriented CF. Different from pointwise CF where each user is represented by a set of rating scores on rated items, in pairwise CF, each user is represented as a set of pairwise preferences over rated items. Then they predict each user's pairwise ordering of items by aggregating the pairwise preferences rather than ratings of similar users to produce a recommendation list. The following example illustrates the representation of the user in pairwise CF.

EXAMPLE 1.2. *Suppose user u_1 has assigned 4, 3, 2 to items i_1, i_2 and i_3 , respectively. The pairwise preferences of the user u include $i_1 \succ i_2$, $i_1 \succ i_3$ and $i_2 \succ i_3$.*

In the pairwise CF, the similarity between two users is determined by their pairwise rankings of the items. A commonly used similarity measure is the Kendall's τ correlation coefficient [Kendall 1938; Marden 1996]. Although pairwise ranking-oriented CF has demonstrated significant improvement over previous algorithms in terms of ranking accuracy, it suffers from high computational complexity. Suppose there are N items, and each user has rated n items on average. The average number of commonly rated items for each pair of users is \bar{n} , and the number of the neighbor users is l . Then the time complexity of computing the correlation between each pair of users is $\mathcal{O}(n + \bar{n}^2)$, and the time complexity of the ranking prediction procedure is $\mathcal{O}(l(N - n)^2)$ for each user. In most cases, the majority of the items are unrated for each user, resulting in a very large value of $N - n$. As a consequence, the computations of the pairwise ranking-oriented CF algorithms are usually much more expensive than the pointwise rating-oriented CF algorithms.

Beside the computational complexity, there could be problems with the accuracy of pairwise ranking-oriented CF algorithms too. First, Kendall's τ correlation coefficient ignores the ranking positions of the items, which may lead to inaccurate similarities when discovering similar users. Second, Kendall's τ correlation coefficient cannot handle the equal preferences tie, which may cause information loss in training and prediction. Third, pairwise ranking-oriented CF algorithms attempt to predict relative preferences between pair of items rather than the total rankings, which may also result in accuracy loss due to rank aggregation.

EXAMPLE 1.3. *Suppose users $\{u_1, u_2, u_3\}$ have assigned ratings of $\{4, 3, 2\}$, $\{4, 2, 3\}$, $\{3, 4, 2\}$ to items $\{i_1, i_2, i_3\}$ respectively. The Kendall's τ correlation coefficients $\tau(u_1, u_2) = \tau(u_1, u_3) = \frac{1}{3}$, indicating that u_2 and u_3 are equally similar to u_1 . However, considering the position information, u_2 is more similar to u_1 than u_3 , as u_2 and u_1 share a same most favorite item i_1 .*

1.3. Ranking-Oriented CF: The Listwise CF

To tackle the aforementioned problems with the pairwise CF algorithms, we propose ListCF, a *listwise* memory-based ranking-oriented CF algorithm, which can reduce the time complexity of both the training and prediction phases while maintaining or even improving the prediction accuracy as compared to conventional pairwise ranking-oriented CF algorithms. ListCF can directly predict a ranking list of items for the target user based on the probability distributions over the permutations of items and thus avoids the accuracy loss caused by the prediction and aggregation of pairwise preferences.

In particular, ListCF utilizes the Plackett-Luce model [Marden 1996], a widely used permutation probability model in the literature of learning to rank, to represent each user as a probability distribution over the permutations of rated items. In this way, both the ranking positions and equal preferences can be well considered. In the permu-

tation probability model, it is assumed that more “correct” permutations where items with larger ranking scores are ranked higher are assigned with larger probabilities. Since the number of permutations on a set of n items is of order $\mathcal{O}(n!)$, the computation costs might be too high to calculate. In ListCF, we group these permutations into groups, where each group of permutations share same items in their top- k positions ($1 \leq k \leq n$). Then we only consider the probabilities of the the top- k permutation groups for significant improvement in efficiency. In doing this, given a set of n items, the number of probabilities is $\frac{n!}{(n-k)!}$, which is much smaller than $n!$. In special case of $k = 1$, there are only n top-1 permutation groups with different top-1 items. Actually the top-1 probability has been successfully applied in the learning to rank problem [Cao et al. 2007].

In ListCF, the similarity between two users is measured based on the Kullback–Leibler (KL) divergence [Kullback 1997] between their top- k probability distributions over the set of co-rated items. Then for each user, those users with higher similarity scores are selected as the set of neighborhood users. Given a target user and the neighborhood users, ListCF infers predictions by minimizing the cross entropy loss between their probability distributions over permutations of items with gradient descent.

1.4. Incremental ListCF

In online recommender systems, the data is constantly changing as users provide new feedbacks or modify existing preferences at any time. A straightforward way is to periodically recalculate the similarity between users or items in an offline way [Yang et al. 2012]. However, since the similarity calculation process is very time-consuming, it is too inefficient with frequent recalculations, whereas the recommendation results might be inaccurate and unresponsive with long intervals between recalculations. Following the prior works [Papagelis et al. 2005; Miranda and Jorge 2009; Yang et al. 2012], we design incremental update strategies to make ListCF more scalable and adaptive to the constantly changing data.

We perform extensive experiments on benchmark datasets to validate the performance of our proposed approaches in comparison with state-of-the-art algorithms. ListCF achieves sharply efficiency gains in comparison with pairwise CF, and it also demonstrates significant improvement in recommendation accuracy over state-of-the-art memory-based and most model-based CF algorithms. Besides, our incremental ListCF also shows extremely promising performance gains in efficiency, which can reduce the computation costs from $O(\bar{n})$ to $O(1)$, where \bar{n} is the average number of items rated by users.

Contributions. We make the following contributions.

- (1) We investigate the listwise paradigm of memory-based CF, which seeks improvement in both accuracy and efficiency in comparison with pairwise CF.
- (2) We propose ListCF, a listwise memory-based ranking-oriented CF algorithm, based on the listwise probability representation and optimization.
- (3) We utilize matrix-based formulations to interpret the three paradigms of memory-based CF, i.e., pointwise, pairwise and listwise CF, and reveal insightful connections among them.
- (4) We present the incremental updating strategies in ListCF which can incrementally update similarity between each pair of users when someone submits a new rating or updates an existing rating.

Organization. The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 presents the preliminaries. Section 4 formulates the listwise

collaborative filtering paradigm and presents the details of our ListCF algorithm. Section 5 presents the incremental ListCF method which updates user-user similarities incrementally. Section 6 discusses the connections among the paradigms of pointwise, pairwise and listwise CF. Section 7 introduces the experimental settings. Section 8 reports the experimental results, and section 9 concludes the paper.

2. RELATED WORK

2.1. Recommender Systems

In recent years, many recommendation algorithms have been proposed, which fall into content-based, collaborative filtering (CF) and hybrid algorithms. Content-based algorithms make recommendations based on similarities between the vector representing the user profile and the vectors representing the items, where the vectors for user profile and items are weighted explicit features [Lops et al. 2011]. CF avoids the necessity of collecting extensive content information about items and users by fully utilizing the user-item rating matrix to make recommendations. Thus, CF can be easily adopted in different recommender systems without requiring additional domain knowledge [Liu and Yang 2008]. Given the effectiveness and convenience, many CF algorithms have been proposed, which are either rating-oriented CF or ranking-oriented CF. In addition, CF algorithms can easily integrate various additional information like social networks [Jamali and Ester 2010; Jiang et al. 2012], reviews [Zhang et al. 2014] and tweets [Ren et al. 2013] to make more accurate recommendations [Shi et al. 2014; Huang et al. 2015a].

2.2. Rating-oriented CF

Traditional rating-oriented CF predicts the absolute scores of individual users to unrated items, and thus we refer to it as *pointwise* CF. As a classical CF paradigm, rating-oriented CF has been extensively investigated, where most algorithms are either memory-based [Resnick et al. 1994; Herlocker et al. 1999; Sarwar et al. 2001; Deshpande and Karypis 2004] or model-based [Si and Jin 2003; Vucetic and Obradovic 2005; Shani et al. 2005; Jiang et al. 2012; Tang et al. 2013].

In memory-based rating-oriented CF, two representatives are user-based CF [Herlocker et al. 2002] and item-based CF [Linden et al. 2003; Sarwar et al. 2001], which make predictions by utilizing historical ratings associated with similar users or similar items. The commonly used similarity metrics are Pearson correlation coefficient [Herlocker et al. 2002] and cosine similarity [Breese et al. 1998]. Resulting from the robustness and easy implementation, many commercial systems such as Amazon.com belong to this category [Linden et al. 2003].

Model-based rating-oriented CF learns a model based on the observed ratings to make rating predictions. Latent factor models such as matrix factorization (MF) [Koren et al. 2009] that uses dimensionality reduction techniques have become important research directions in this area. A variety of adaptations of the MF model have demonstrated promising prediction performance, such as Probabilistic Matrix Factorization (PMF) [Mnih and Salakhutdinov 2007], Non-negative Matrix Factorization (NMF) [Lee and Seung 2000], Max-Margin Matrix Factorization (MMMF) [Rennie and Srebro 2005], SVDFeature [Chen et al. 2012] and Factorization machines (FM) [Rendle 2012].

2.3. Ranking-oriented CF

Recently, the formulation of recommendation problem is shifting away from rating-oriented to ranking oriented [McNee et al. 2006]. Different from rating-oriented CF, ranking-oriented CF directly predicts a preference ranking of items for each user.

Similarly, ranking-oriented CF algorithms can also be categorized into memory-based ranking-oriented CF [Liu and Yang 2008; Wang et al. 2012b; 2014] and model-based ranking-oriented CF [Koren and Sill 2011; Rendle et al. 2009; Shi et al. 2010].

EigenRank [Liu and Yang 2008] is a classic memory-based ranking-oriented CF algorithm, which measures the similarity between users based on the correlation between their pairwise rankings of the co-rated items and ranks items by the preferences of similar users. VSRank [Wang et al. 2012b; 2014] adopts the vector space model in representing user pairwise preferences and makes further improvement by considering the relative importance of each pairwise preference. Resulting from the pairwise preference based representations and predictions, we refer to EigenRank and VSRank as *pairwise* memory-based ranking-oriented CF algorithms. Although they achieve better performance in recommendation accuracy than rating-oriented CF, they suffer from serious efficiency problem. In this paper, we propose a *listwise* memory-based ranking-oriented CF algorithm to reduce the computational complexity while maintaining or even improving the ranking performance.

A variety of model-based ranking-oriented CF algorithms have been presented by optimizing ranking-oriented objective functions, e.g., bayesian personalized ranking (BPR) [Rendle et al. 2009], CLiMF [Shi et al. 2012], CofiRank [Weimer et al. 2007], and ListRank-MF [Shi et al. 2010]. BPR and CLiMF model the binary relevance data and optimize binary relevance metrics, i.e., Area Under the Curve (AUC) in BPR and Mean Reciprocal Rank (MRR) in CLiMF, which are not suited for graded relevance data. CofiRank [Weimer et al. 2007] minimizes a convex upper bound of the Normalized Discounted Cumulative Gain (NDCG) [Järvelin and Kekäläinen 2000; Järvelin and Kekäläinen 2002] loss through matrix factorization. In addition, it is natural to apply learning to rank models in ranking-oriented model-based recommender systems. For example, ListRank-MF [Shi et al. 2010] integrates the listwise learning to rank technique into the probabilistic matrix factorization model for recommendation. In [Kahng et al. 2011], a context-aware learning to rank algorithm is proposed that incorporates several context features, such as time and weather, into the ranking model.

In this paper, we focus on memory-based CF recommendation algorithms and propose a listwise CF algorithm ListCF, which demonstrates significant advantages in recommendation efficiency and accuracy.

2.4. Incremental CF

In real-world recommender systems, it is common that the data is gradually accumulated as users interact with items, whilst users expect to receive updated recommendation results once they provide new feedbacks. Therefore, it is necessary that recommender systems are able to incrementally evolve the prediction model from the constantly updated data.

There are several relevant pioneering works in the field of incremental CF recommendation. For memory-based CF, Papagelis et al. proposed a user-based incremental algorithm, which incrementally updates similarity between users when a user submits a new rating or updates an existing rating [Papagelis et al. 2005]. Based on this work, Miranda et al. presented an item-based incremental algorithm to incrementally update similarity between items for binary ratings [Miranda and Jorge 2009], and Yang et al. [Yang et al. 2012] provided an item-based incremental algorithm for multiple-level ratings. For model-based CF, Sarwar et al. proposed an incremental algorithm for Singular Value Decomposition (SVD)-based CF algorithm based on the fold-in technique [Sarwar et al. 2002]. Luo et al. implemented an MF-based incremental recommendation algorithm, which can not only deal with the new users/items, but also update the already trained latent factors on arrival of corresponding feedbacks [Luo et al. 2012].

All of these efforts are proposed to speed up rating-oriented CF algorithms. In this paper, we provide incremental ListCF to accelerate the listwise ranking-oriented CF algorithm.

3. PRELIMINARIES

Let U be a set of users and I be a set of items. Let R be a user-item rating matrix, in which each element $r_{u,i}$ is the rating of user u to item i . For each user $u \in U$, the set of items rated by u is represented as $I_u \subseteq I$. For each pair of users u and v , the set of items commonly rated by both u and v is denoted as $I_{u,v} = I_u \cap I_v$. The conventional collaborative filtering (CF) algorithms can be formulated as follows.

3.1. Pointwise CF

Rating-oriented CF, referred to as pointwise CF, recommends items to users based on the rating values predicted by aggregating the ratings associated with similar users. The most commonly used similarity metric is Pearson correlation coefficient $\rho(u, v)$ [Herlocker et al. 2002]. In practice, the similarity between users u and v can be calculated as shown in Equation (1):

$$\rho(u, v) = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \bar{r}_v)^2}}, \quad (1)$$

where \bar{r}_u and \bar{r}_v are the average ratings of u and v respectively.

For each target user u , a set of neighborhood users $N_u \subset U$ are selected according to the descending order of the similarities between u and others. The rating value of the user u to an unrated item t can be predicted by weighting the ratings of his neighborhood users N_u^t who have rated the item t , which is shown in Equation (2):

$$\hat{r}_{u,t} = \bar{r}_u + \frac{\sum_{v \in N_u^t} \rho(u, v)(r_{v,t} - \bar{r}_v)}{\sum_{v \in N_u^t} \rho(u, v)} \quad (2)$$

3.2. Pairwise CF

Pairwise CF recommends items to users based on the pairwise preferences over pairs of rated items from similar users. The similarity between two users is determined by their pairwise preferences over the co-rated items. The most commonly used similarity metric between two users u and v is Kendall's τ correlation coefficient [Marden 1996] $\tau_{u,v}$, which is shown in Equation (3):

$$\tau(u, v) = \frac{N_c - N_d}{\frac{1}{2}n(n-1)}, \quad (3)$$

where n is the number of co-rated items, N_c and N_d are the numbers of the concordant pairs and discordant pairs respectively. A pair of items i and j is discordant if i is ranked higher than j in one ranking but lower in the other.

For the target user u , the preference on a pair of unrated items (i, j) can be predicted with a preference function $\Psi_u(i, j)$, which is defined in Equation (4):

$$\Psi_u(i, j) = \frac{\sum_{v \in N_u^{i,j}} \tau(u, v) p_{v,(i,j)}}{\sum_{v \in N_u^{i,j}} \tau(u, v)}, \quad (4)$$

where $N_u^{i,j}$ is the neighborhood users of u who have rated both items i and j , and $p_{u,(i,j)} \in \{-1, 1\}$ is the preference of u on the pair of items (i, j) . In particular, the value for the preference $p_{u,(i,j)}$ can be represented as follows:

$$p_{u,(i,j)} = \begin{cases} -1, & \text{if } r_{u,i} \leq r_{u,j} \\ 1, & \text{if } r_{u,j} > r_{u,i} \end{cases} \quad (5)$$

Based on Equation (4), the more often the users in N_u prefer item i than item j , the stronger evidence for $\Psi_u(i, j) > 0$. After predicting the pairwise preferences on unrated items for each user, a total ranking of items can be obtained by applying a preference aggregation method.

4. LISTCF: LISTWISE CF

In this section, we propose ListCF, a novel paradigm of listwise ranking-oriented collaborative filtering (CF) algorithm based on the memory-based CF framework. We firstly introduce the listwise representation of the ListCF based on the Plackett-Luce model [Marden 1996], and then present the two phases of the memory-based CF algorithms:

- **Phase I: Similarity Calculation.** It calculates the similarities between pairs of users, based on which a set of neighborhood users can be discovered for each user.
- **Phase II: Ranking Prediction.** Based on the neighborhood users, phase II predicts a ranking list of items for each user by aggregating preferences of neighborhood users for recommendation purpose.

4.1. Representation

In ListCF, we utilize Plackett-Luce model [Marden 1996], a widely used permutation probability model, to represent each user as a probability distribution over the permutations of rated items. It assumes that given a ranking function on a set of items, any permutation on the item set is possible. However, different permutations may have different probabilities, and more “correct” permutations where items with larger ranking scores are ranked higher should be assigned with larger probabilities.

Let I be a set of items. A permutation of I is a bijection from I to itself and it can be represented as an ordered list $\pi = \langle \pi_1, \pi_2, \dots, \pi_n \rangle$, where $\pi_i \in I$ denotes the item at position i and $\pi_i \neq \pi_j$ if $i \neq j$. The set of all the possible permutations of I is denoted as Ω^I .

DEFINITION 4.1 (PROBABILITY OF PERMUTATION). *Let $\phi(\cdot)$ be an increasing and strictly positive function. Given a permutation $\pi = \langle \pi_1, \pi_2, \dots, \pi_n \rangle$ and the ratings of the items $\{r_{\pi_1}, r_{\pi_2}, \dots, r_{\pi_n}\}$, the probability of π is defined as follows:*

$$P(\pi) = \prod_{i=1}^n \frac{\phi(r_{\pi_i})}{\sum_{k=i}^n \phi(r_{\pi_k})}.$$

For a set of n items, the number of different permutations is $n!$. Thus, it is too time-consuming to calculate the probabilities of all the permutations. To cope with this problem, an alternative solution is the *top-k probability model* [Cao et al. 2007] ($1 \leq k \leq n$), which only focuses on the permutations of items within the top- k positions.

DEFINITION 4.2 (TOP- k PERMUTATION SET). *Given a set of items I , the top- k permutation set $\mathcal{G}_k(i_1, i_2, \dots, i_k)$ of I is a set containing all the permutations of I which share the same top- k items, i.e., i_1, i_2, \dots, i_k where $i_1, i_2, \dots, i_k \in I$. Formally,*

$$\mathcal{G}_k(i_1, i_2, \dots, i_k) = \{\pi \mid \pi \in \Omega^I, \pi_j = i_j, j = 1, \dots, k\}.$$

Two top- k permutation sets of I are different if the items in the top- k positions are different. Let \mathcal{G}_k^I denote the set of different top- k permutation sets for I . Please note that the cardinality of $\mathcal{G}_k(i_1, i_2, \dots, i_k)$ is $(n-k)!$, and the cardinality of \mathcal{G}_k^I is $n!/(n-k)!$.

EXAMPLE 4.1. Let $I = \{i_1, i_2, i_3\}$. There are totally $3! = 6$ possible permutations in Ω^I : $\langle i_1, i_2, i_3 \rangle$, $\langle i_1, i_3, i_2 \rangle$, $\langle i_2, i_1, i_3 \rangle$, $\langle i_2, i_3, i_1 \rangle$, $\langle i_3, i_1, i_2 \rangle$, and $\langle i_3, i_2, i_1 \rangle$. There are merely $3!/2! = 3$ top-1 permutation sets: $\mathcal{G}_1^I = \{\mathcal{G}_1(i_1), \mathcal{G}_1(i_2), \mathcal{G}_1(i_3)\}$, where $\mathcal{G}_1(i_1) = \{\langle i_1, i_2, i_3 \rangle, \langle i_1, i_3, i_2 \rangle\}$, $\mathcal{G}_1(i_2) = \{\langle i_2, i_1, i_3 \rangle, \langle i_2, i_3, i_1 \rangle\}$, and $\mathcal{G}_1(i_3) = \{\langle i_3, i_1, i_2 \rangle, \langle i_3, i_2, i_1 \rangle\}$.

Based on the Definition 4.2, the probability of the top- k permutation $P(\mathcal{G}_k(i_1, i_2, \dots, i_k))$ equals to the sum of the probabilities of permutations in which the items i_1, i_2, \dots, i_k are ranked in the top- k positions. The number of top- k permutation sets in \mathcal{G}_k^I is $n!/(n-k)!$, with each top- k permutation set containing $(n-k)!$ permutations of I . Therefore it still has to calculate $n!/(n-k)! \cdot (n-k)! = n!$ probabilities of permutations in total. In [Cao et al. 2007], an efficient way to calculate the probability of the top- k permutation set $P(\mathcal{G}_k(i_1, i_2, \dots, i_k))$ is shown in Lemma 4.1.

LEMMA 4.1. *The probability of the top- k permutation set $P(\mathcal{G}_k(i_1, i_2, \dots, i_k))$ equals to the probability of that the items i_1, i_2, \dots, i_k are ranked in the top- k positions respectively. Formally,*

$$P(\mathcal{G}_k(i_1, i_2, \dots, i_k)) = \prod_{j=1}^k \frac{\phi(r_{\pi_j})}{\sum_{l=j}^n \phi(r_{\pi_l})}, \quad \forall j = 1, \dots, k : \pi_j = i_j, \quad (6)$$

where r_{π_l} is the rating of the item which is ranked in position l , $l = 1, 2, \dots, n$.

Especially, when $k = 1$, the top-1 probability of item $i (i \in I)$ is:

$$P(\mathcal{G}_1(i)) = \frac{\phi(r_{\pi_1})}{\sum_{j=1}^n \phi(r_{\pi_j})}, \quad \pi_1 = i.$$

According to [Cao et al. 2007], the probabilities of the top- k permutation sets in \mathcal{G}_k^I form a probability distribution, i.e., for each $g \in \mathcal{G}_k^I$, we have $P(g) > 0$, and $\sum_{g \in \mathcal{G}_k^I} P(g) = 1$. In this study, we refer to the probability distribution over the top- k permutation sets as the *top-k probability model*, which uses the top- k permutation sets instead of the full permutations of all the items.

4.2. Phase I: Similarity Calculation

In ListCF, we define the similarity between two users based on their probability distributions over the top- k permutation sets of co-rated items. In this paper, we use the exponential function $\phi(r_i) = e^{r_i}$ as our ranking function in Equation (6), as the exponential function would emphasize the importance of the highly rated items.

In particular, given two users u and v , let I_u and I_v be the sets of items rated by users u and v respectively and $I_{u,v} = I_u \cap I_v$ be the set of their co-rated items. Then

$\mathcal{G}_k^{I_{u,v}}$ represents the set of top- k permutation sets of $I_{u,v}$. We use P_u and P_v to denote the probability distributions of u and v over $\mathcal{G}_k^{I_{u,v}}$ according to the top- k probability model, which can be calculated with Equation (6) based on their ratings. Thus the similarity between user u and v can be estimated as the similarity between the probability distributions of P_u and P_v .

In this study, we use Kullback-Leibler (KL) divergence [Kullback 1997] based metric for similarity calculation, since it is a common measure of the difference between two probability distributions. In probability theory and information theory, KL divergence has been widely used in machine learning [Challis and Barber 2013], image restoration [Seghouane 2011], vision recognition [Wang et al. 2012a] and speech synthesis [Ling and Dai 2012]. Given two users u and v , the KL divergence of P_u from P_v is defined as follows:

$$D_{KL}(P_v \| P_u) = \sum_{g \in \mathcal{G}_k^{I_{u,v}}} P_v(g) \log_2 \left(\frac{P_v(g)}{P_u(g)} \right),$$

where $P_u(g)$ and $P_v(g)$ is the probability of the top- k permutation set $g \in \mathcal{G}_k^{I_{u,v}}$ for u and v respectively. $D_{KL}(P_u \| P_v)$ can be calculated in the similar way. Obviously, the KL divergence is asymmetric, i.e., $D_{KL}(P_v \| P_u) \neq D_{KL}(P_u \| P_v)$. In this study, we define a symmetric similarity metric based on the KL divergence, which is shown as follows:

$$s(u, v) = 1 - \frac{1}{2} [D_{KL}(P_v \| P_u) + D_{KL}(P_u \| P_v)]. \quad (7)$$

According to the similarity metric in Equation (7), the users with few commonly rated ratings trend to obtain high similarity. Like in [Herlocker et al. 2002], this can be alleviated by multiplying the similarity function with $\min\{|I_{u,v}|/c, 1\}$, where c is a threshold. When the number of commonly rated items falls below this threshold, the similarity is scaled down.

Since $D_{KL}(P_v \| P_u), D_{KL}(P_u \| P_v) \geq 0$, according to Equation (7), the similarity between each pair of users is no greater than 1. Formally, $\forall u, v \in U : s(u, v) \leq 1$. The similarity between u and v reaches the maximum bound ($s(u, v) = 1$) only if their probability distributions P_u and P_v over the top- k permutation sets are the same. We also note that $s(u, v)$ may be negative and has no lower bound. However, this is not an important issue in our model because we are just interested in discovery of limited number of users with high similarities to the target user as the neighborhood users. The similarity calculation can be illustrated in Example 4.2.

EXAMPLE 4.2. Let $U = \{u_1, u_2, u_3\}$ be the set of users, and $I = \{i_1, i_2, i_3\}$ be the set of items. Suppose that u_1 has assigned ratings of $\{5, 3, 4\}$, u_2 has assigned ratings of $\{5, 4, 3\}$, and u_3 has assigned ratings of $\{4, 3, 5\}$ to the items.

In ListCF, for users u_1, u_2 and u_3 , there are $\frac{3!}{(3-1)!} = 3$ top-1 permutation sets with different top-1 items of i_1, i_2 and i_3 respectively. The probability distributions over the set of the top-1 permutations for u_1, u_2 and u_3 are shown as follows:

$$\begin{aligned} P_{u_1} &= \left(\frac{e^5}{e^4 + e^3 + e^5}, \frac{e^3}{e^4 + e^3 + e^5}, \frac{e^4}{e^4 + e^3 + e^5} \right) = (0.665, 0.090, 0.245), \\ P_{u_2} &= \left(\frac{e^5}{e^4 + e^3 + e^5}, \frac{e^4}{e^4 + e^3 + e^5}, \frac{e^3}{e^4 + e^3 + e^5} \right) = (0.665, 0.245, 0.090), \\ P_{u_3} &= \left(\frac{e^4}{e^4 + e^3 + e^5}, \frac{e^3}{e^4 + e^3 + e^5}, \frac{e^5}{e^4 + e^3 + e^5} \right) = (0.245, 0.090, 0.665). \end{aligned}$$

Thus according to Equation (7), the similarities are $s(u_1, u_2) = 0.776$, $s(u_1, u_3) = 0.395$ and $s(u_2, u_3) = -0.052$.

4.3. Phase II: Ranking Prediction

For a target user u , let N_u be the set of neighborhood users and $T_u = \{t_1, t_2, \dots, t_p\}$ be the set of items to be predicted. The ultimate goal of ranking-oriented CF is to predict a preference ranking of items in T_u for u .

Let \hat{P}_u be the probability distribution over the set of top- k permutation sets $\mathcal{G}_k^{T_u}$ to be predicted for u . In order to guarantee that the sum of the probabilities in \hat{P}_u is equal to 1, for each top- k permutation set $\forall g \in \mathcal{G}_k^{T_u}$, the probability $\hat{P}_u(g)$ is denoted as follows:

$$\hat{P}_u(g) = \frac{\varphi_{u,g}}{\sum_{g' \in \mathcal{G}_k^{T_u}} \varphi_{u,g'}}, \quad (8)$$

where $\{\varphi_{u,g} | \forall g \in \mathcal{G}_k^{T_u}\}$ are unknown variables to be predicted for user u and $\sum_{g \in \mathcal{G}_k^{T_u}} \hat{P}_u(g) = 1$ holds.

In this study, following the same assumption of memory-based CF, we make predictions based on a set of neighborhood users with similar ranking preferences to the target user. If users have similar ranking preferences in the past, they are most likely to have similar ranking preferences in the future, which means that \hat{P}_u should be as close as possible to the probability distributions of the neighborhood users over their sets of top- k permutation sets.

We use the cross entropy loss function to make predictions, as it is widely-used for optimizing similarity/distance between probability distributions in machine learning, and it has been successfully used in text mining [Cao et al. 2007], multimedia retrieval [Li et al. 2011], and pattern recognition [Tegmeyer et al. 2014] fields.

Given a target user u with a set of unrated items T_u and a neighborhood user $v \in N_u$, let I_v be the set of items that the neighborhood user v has rated, and $T_{u,v} = T_u \cap I_v$. Thus $\mathcal{G}_k^{T_{u,v}}$ is the set of top- k permutation sets of $T_{u,v}$. Let \hat{P}'_u and P'_v be the respective probability distributions of u and v over $\mathcal{G}_k^{T_{u,v}}$. The cross entropy between \hat{P}'_u and P'_v can be calculated as follows:

$$E(\hat{P}'_u, P'_v) = - \sum_{g \in \mathcal{G}_k^{T_{u,v}}} P'_v(g) \log_2(\hat{P}'_u(g)). \quad (9)$$

Please note that in the above equation, P'_v is different from P_v in the similarity calculation phase, as they are based on different sets of items. P'_v is based on $T_{u,v}$ while P_v is based on the co-rated items $I_{u,v} = I_u \cap I_v$ of user u and v . Obviously $\sum_{g \in \mathcal{G}_k^{T_{u,v}}} P'_v(g) = 1$ holds. Besides, \hat{P}_u is the probability distribution over $\mathcal{G}_k^{T_u}$, while \hat{P}'_u is the probability distribution over $\mathcal{G}_k^{T_{u,v}}$ with a specific neighborhood user $v \in N_u$.

ListCF attempts to make predictions by minimizing the weighted sum of the cross entropy loss between the probability distributions of the target user and his neighborhood users over the set of the top- k permutation sets of $T_{u,v}$. For a target user u and the set of unrated items T_u , the cross entropy-based loss function is:

$$\begin{aligned} \arg \min_{\varphi_u} \quad & \sum_{v \in N_u} s(u, v) \cdot E(\hat{P}'_u, P'_v) \\ \text{s.t.} \quad & \forall g \in \mathcal{G}_k^{T_u} : \varphi_{u,g} \geq 0. \end{aligned} \quad (10)$$

We optimize Equation (10) with the gradient descent method. Let

$$F(\varphi_u) = \sum_{v \in N_u} s(u, v) \cdot E(\hat{P}'_u, P'_v) \quad (11)$$

The derivative of F with respect to $\varphi_{u,g}$ is:

$$\frac{\partial F}{\partial \varphi_{u,g}} = \sum_{v \in N_u} \frac{s(u, v)}{\ln 2 \cdot \sum_{g' \in \mathcal{G}_k^{T_u, v}} \varphi_{u,g'}} - \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\ln 2 \cdot \varphi_{u,g}}. \quad (12)$$

The full arithmetic derivations of the gradients can be found in Appendix A. Given a learning rate η , the update formula of the gradient descent method is:

$$\varphi_{u,g} \leftarrow \varphi_{u,g} - \eta \frac{\partial F}{\partial \varphi_{u,g}}. \quad (13)$$

After updating $\varphi_{u,g}$, we can easily get the probabilities $\hat{P}_u(g)$ according to Equation (8), which meets the constraint of $\sum_{g \in \mathcal{G}_k^{T_u}} \hat{P}_u(g) = 1$.

In order to obtain an ultimate ranking of T_u for each user u , the probabilities of the top- k permutation sets need to be aggregated. One intuitive way is to get the probabilities of the top-1 permutation sets and rank the items in T_u according to the descending order of the corresponding probabilities. The probability of the top-1 permutation set $P(\mathcal{G}_1^{T_u}(t_j))$ actually equals to the sum of the probabilities of the top- k ($k > 1$) permutation sets with the item $t_j \in T_u$ ranked in the top-1 position. Here gives an example.

EXAMPLE 4.3. Suppose user u has items $I = \{i_1, i_2, i_3\}$ to be recommended. We have probability distribution $\{0.2, 0.3, 0.1, 0.1, 0.1, 0.2\}$ over the set of top-2 permutation sets $\mathcal{G}_2^I = \{\mathcal{G}_2(i_1, i_2), \mathcal{G}_2(i_1, i_3), \mathcal{G}_2(i_2, i_1), \mathcal{G}_2(i_2, i_3), \mathcal{G}_2(i_3, i_1), \mathcal{G}_2(i_3, i_2)\}$. We can obtain the probabilities of that each item is ranked in the top-1 position: $P(\mathcal{G}_1(i_1)) = 0.2 + 0.3 = 0.5$, $P(\mathcal{G}_1(i_2)) = 0.1 + 0.1 = 0.2$, and $P(\mathcal{G}_1(i_3)) = 0.1 + 0.2 = 0.3$. Since $P(\mathcal{G}_1(i_1)) > P(\mathcal{G}_1(i_3)) > P(\mathcal{G}_1(i_2))$, the recommendation list is ordered as $\langle i_1, i_3, i_2 \rangle$.

4.4. Pseudocode and Discussion

The pseudocode of the ListCF algorithm is shown in Algorithm 1, where lines 1-7 calculate the similarities between users and discover the neighbor users for each target user, and lines 8-23 predict the ranking of items to make recommendations. When $k > 1$ in the top- k probability model, lines 19-21 aggregate the probabilities of the top- k permutation sets to obtain the probabilities of the top-1 permutation sets, that are the probabilities of each item $t \in T_u$ ranked in the first position.

In the proposed ListCF model, we use the top- k ($k \geq 1$) probability model [Cao et al. 2007] in the similarity calculation and ranking prediction phases. ListCF with different values of k may achieve different recommendation performances. On one hand, generally the accuracy will be higher when a larger k value is used. On the other hand, the computational complexity of similarity calculation and ranking prediction procedures will be exponentially increased. In particular, for a user u associated with n items, the number of permutation sets in the top- k probability model is $n!/(n-k)!$, and specially this number is merely n when $k = 1$.

Generally speaking, when k increases, the accuracy could be somehow improved, but the efficiency is decreased sharply. For example, the time complexity of ListCF based on the top- k ($k > 1$) probability model is $(n-1)!/(n-k)!$ times higher than that of ListCF based on the top-1 probability model. Thus we set $k = 1$ when compare ListCF

ALGORITHM 1: The ListCF Algorithm

Input: An item set I , a user set U , and a rating matrix $R \in \mathbb{R}^{M \times N}$. A set of rated items $I_u \subseteq I$ by each user $u \in U$. The maximal number of iterations $maxIteration$ and error threshold ϵ .

Output: A ranking $\hat{\tau}_u$ of items for each user $u \in U$.

```

1 for  $u \in U$  do
2   for  $v \in U$  and  $u \neq v$  do
3      $P_u, P_v \leftarrow \text{TopKProDist}(I_u, I_v, R)$  /* Eq.6 */
4      $sim(u, v) \leftarrow \text{Similarity}(P_u, P_v)$  /* Eq.7 */
5   end
6    $N_u \leftarrow \text{SelectNeighbors}(\{sim(u, v)\}_{v \in U/u})$ 
7 end
8 for  $u \in U$  do
9    $t = 1$ 
10  repeat
11     $\varepsilon = 0$ 
12     $\text{Initialize}(\varphi_u^0)$ 
13    for  $g \in \mathcal{G}_k^{T_u}$  do
14       $\varphi_{u,g}^t \leftarrow \text{Update}(N_u, sim, R)$  /* Eq.13 */
15       $\varepsilon += \sqrt{\sum (\varphi_{u,g}^t - \varphi_{u,g}^{t-1})^2}$ 
16    end
17     $t \leftarrow t + 1$ 
18  until  $t > maxIteration$  or  $\varepsilon < \epsilon$ ;
19  for  $t \in T_u$  do
20     $P(t) \leftarrow \text{Aggregation}(\{\varphi_{u,g}\}_{g \in \mathcal{G}_k^{T_u}})$ 
21  end
22   $\hat{\tau}_u \leftarrow \text{Ordering}(\{P(t)\}_{t \in T_u})$ 
23 end

```

with other recommendation algorithms in the experimental section. The experimental results reported in Section 8.1 also illustrate our analysis.

4.5. Discussion

In comparison with conventional pairwise ranking-oriented CF, listwise CF demonstrates many advantages in efficiency (when $k = 1$) and accuracy on account of the listwise representation.

On the efficiency side, let M and N be the number of users and items. Suppose each user has rated n items on average, and the number of commonly rated items for each pair of users is \bar{n} . In the similarity calculation phase, the time complexity of ListCF is $\mathcal{O}(M^2(n + \bar{n}))$. Recall that the complexity of pairwise CF is $\mathcal{O}(M^2(n + \bar{n}^2))$, which is $(n + \bar{n}^2)/(n + \bar{n})$ times higher than that of ListCF. Suppose each user has l neighbor users on average, and each prediction process maximally performs d iterations for gradient descent optimization. In the ranking prediction phase, the time complexity of ListCF is $\mathcal{O}(M(N - n)ld)$, where $N - n$ is the number of items to recommend. Recall that the complexity of pairwise CF is $\mathcal{O}(M(N - n)^2l)$, which is $(N - n)/d$ times higher than that of ListCF, and $d \ll N - n$ in our experiments. Since d is a constant, the ratio of the complexity of pairwise CF to that of ListCF increases linearly with the growth of the number of items to be predicted in the ranking prediction phase.

On the accuracy side, firstly, permutation probability based similarity metric considers the ranking position information of items, which can discover more accurate neigh-

bor users with similar preferences. Secondly, permutation probability based representation can easily handle the the case of equal preferences on items, i.e., assigns equal probability to the permutations in which the top items have equal ranking scores. Thirdly, ListCF attempts to make top- n recommendation by optimizing listwise loss function, which is more natural and accurate. The following example demonstrates the accuracy loss of the Kendall's τ correlation coefficient because of not considering the ranking position information of items.

EXAMPLE 4.4. *Consider Example 4.2, where three users $\{u_1, u_2, u_3\}$ have assigned ratings of $\{5, 3, 4\}$, $\{5, 4, 3\}$, and $\{4, 3, 5\}$ to items $\{i_1, i_2, i_3\}$ respectively.*

In pointwise CF, the similarity (Pearson correlation coefficient) between u_1 and u_2 and the similarity between u_1 and u_3 are $\rho(u_1, u_2) = \rho(u_1, u_3) = 0.5$.

In pairwise CF, the similarity (Kendall's τ correlation coefficient) between u_1 and u_2 and the similarity between u_1 and u_3 are $\tau(u_1, u_2) = \tau(u_1, u_3) = 0.333$.

In ListCF, according to the results in Example 4.2, the similarity between u_1 and u_2 is $s(u_1, u_2) = 0.776$ and the similarity between u_1 and u_3 is $s(u_1, u_3) = 0.395$, and thus $s(u_1, u_2) > s(u_1, u_3)$.

Since recommender systems aim to recommend the items that users may be most interested in, we should emphasize the items with highest scores in the similarity calculation and ranking prediction phases, and thus in Example 4.4, similarity between u_2 and u_1 should be higher than that between u_3 and u_1 . Obviously, ListCF can successfully recognize it while pointwise CF and pairwise CF cannot.

5. INCREMENTAL LISTCF

In memory-based collaborative filtering (CF) algorithms, the Phase I (similarity calculation) is very time-consuming, which calculates the similarity between each pair of users. In real-world scenarios, the users' rating data is constantly changing, it is too inefficient with frequently recalculations for accurate and responsive recommendation results. In this section, we present Incremental ListCF, an incremental version of ListCF that allows incrementally updating similarity between users when a new rating is submitted or an existing rating is updated.

Let P_u and P_v be the probability distribution over $\mathcal{G}_k^{I_{u,v}}$ for users u and v respectively. As discussed in Section 4.4, we represent each user with the probability distribution of the top-1 permutations, where the probability of the top-1 permutation for users u and v is:

$$P_u(i) = \frac{\phi(r_{u,i})}{\sum_{j \in I_{u,v}} \phi(r_{u,j})}, \quad P_v(i) = \frac{\phi(r_{v,i})}{\sum_{j \in I_{u,v}} \phi(r_{v,j})}.$$

When a user u submits a new rating or updates an existing rating, the similarity values between him and the rest of the users may need to be recalculated incrementally.

5.1. Case 1: Submission of a New Rating

Suppose user u submits a new rating r_{u,i_a} to item i_a and $i_a \notin I_u$. The similarity between u and any other user v who has rated i_a ($i_a \in I_v$), needs to be updated.

Let $I_{u,v}$ be the set of items that commonly rated by users u and v before u rates i_a . After u submits the new rating r_{u,i_a} to i_a , the commonly rated set of items becomes $I'_{u,v} = I_{u,v} \cup \{i_a\}$. Let P'_u and P'_v be the probability distribution over the commonly rated set of items $I'_{u,v}$ for u and v respectively. The KL-divergence of P'_u from P'_v can be

calculated as follows:

$$D_{KL}(P'_v \| P'_u) = \beta D_{KL}(P_v \| P_u) + \beta \log_2 \frac{\beta}{\alpha} + (1 - \beta) \log_2 \frac{1 - \beta}{1 - \alpha},$$

where

$$\alpha = \frac{\sum_{j \in I_{u,v}} \phi(r_{u,j})}{\sum_{j \in I_{u,v}} \phi(r_{u,j}) + \phi(r_{u,i_a})}, \quad \beta = \frac{\sum_{j \in I_{u,v}} \phi(r_{v,j})}{\sum_{j \in I_{u,v}} \phi(r_{v,j}) + \phi(r_{v,i_a})}.$$

Both the calculation and the formula of the KL-divergence of P'_v from P'_u , $D_{KL}(P'_u \| P'_v)$, is very similar to $D_{KL}(P'_v \| P'_u)$, which is shown as follows:

$$D_{KL}(P'_u \| P'_v) = \alpha D_{KL}(P_u \| P_v) + \alpha \log_2 \frac{\alpha}{\beta} + (1 - \alpha) \log_2 \frac{1 - \alpha}{1 - \beta}.$$

The full arithmetic derivations can be found in Appendix B.1. According to Equation (7), the similarity between u and v is:

$$s(u, v) = 1 - \frac{1}{2} [\alpha D_{KL}(P_u \| P_v) + \beta D_{KL}(P_v \| P_u) + \epsilon], \quad (14)$$

where

$$\begin{aligned} \epsilon &= \alpha \log_2 \frac{\alpha}{\beta} + (1 - \alpha) \log_2 \frac{1 - \alpha}{1 - \beta} + \beta \log_2 \frac{\beta}{\alpha} + (1 - \beta) \log_2 \frac{1 - \beta}{1 - \alpha} \\ &= (\alpha + \beta) \log_2 \frac{\alpha}{\beta} + (\beta - \alpha) \log_2 \frac{1 - \alpha}{1 - \beta}. \end{aligned}$$

Once we calculate α and β , the similarity between u and v can be updated directly by Equation (14) based on the KL-divergence values $D_{KL}(P_u \| P_v)$ and $D_{KL}(P_v \| P_u)$ before u submits the new rating on i_a , without recalculating it with Equation (7).

5.2. Case 2: Update of an existing rating

Suppose user u updated his rating to item i_a from r_{u,i_a} to r'_{u,i_a} . The similarity between u and any other user v who has rated i_a ($i_a \in I_{u,v}$) needs to be recalculated. Since the commonly rated set of items $I_{u,v}$ and the ratings of v have not changed, the probability distribution P_v of user v stays the same. Let the probability distribution of user u changes from P_u to P'_u resulting from the update of the rating from r_{u,i_a} to r'_{u,i_a} by u . The KL-divergence of P'_u from P_v is as follows:

$$D_{KL}(P_v \| P'_u) = D_{KL}(P_v \| P_u) - \log_2 \gamma + P_v(i_a) \log_2 \frac{\gamma P_u(i_a)}{\delta},$$

where

$$\gamma = \frac{\sum_{j \in I_{u,v}} \phi(r_{u,j})}{\sum_{j \in I_{u,v} \setminus \{i_a\}} \phi(r_{u,j}) + \phi(r'_{u,i_a})}, \quad \delta = \frac{\phi(r'_{u,i_a})}{\sum_{j \in I_{u,v} \setminus \{i_a\}} \phi(r_{u,j}) + \phi(r'_{u,i_a})}$$

The KL-divergence of P_v from P'_u can be calculated as follows:

$$D_{KL}(P'_u \| P_v) = \gamma D_{KL}(P_u \| P_v) + \gamma \log_2 \gamma - \gamma P_u(i_a) \log_2 \frac{\gamma P_u(i_a)}{P_v(i_a)} + \delta \log_2 \frac{\delta}{P_v(i_a)}$$

The full arithmetic calculations of the KL-divergences can be found in Appendix B.2. According to Equation (7), the similarity between u and v is:

$$s(u, v) = 1 - \frac{1}{2} [\gamma D_{KL}(P_u \| P_v) + D_{KL}(P_v \| P_u) + \epsilon], \quad (15)$$

where

$$\varepsilon = P_v(i_a) \log_2 \frac{\gamma P_u(i_a)}{\delta} - \gamma P_u(i_a) \log_2 \frac{\gamma P_u(i_a)}{P_v(i_a)} + \delta \log_2 \frac{\delta}{P_v(i_a)} + (\gamma - 1) \log_2 \gamma.$$

Once we calculate γ and δ , the similarity between u and v can be updated directly by Equation (15) based on the KL-divergence values $D_{KL}(P_u||P_v)$ and $D_{KL}(P_v||P_u)$ before u submits the new rating on i_a , without recalculating it with Equation (7).

5.3. Complexity Analysis

Incremental ListCF can significantly reduce the computational costs of the similarity recalculation phase in ListCF when users submit or update some ratings. Let M and N be the number of users and items. Let \bar{n} be the set of the commonly rated items by users u and v . When u submits a new rating or updates the rating to an item i_a that has already been rated by v . In ListCF, the time complexity of recalculating similarity between u and v is $\mathcal{O}(\bar{n})$. In Incremental ListCF, the time complexity of recalculating similarity between u and v is reduced to $\mathcal{O}(1)$ according to Equations (14) and (15).

6. MATRIX-BASED FORMULATIONS OF CF

In this section, we attempt to utilize matrix-based formulations to interpret the three paradigms of memory-based collaborative filtering (CF): pointwise CF, pairwise CF and listwise CF, and try to discover the connections among them.

First of all, we reveal that for CF algorithms, the similarity of the Pearson correlation coefficient between users is equal to their cosine similarity if the rating matrix satisfies certain constraints in Section 6.1. Based on this discovery, we prove that pairwise CF actually performs a pointwise CF algorithm on a user-(item-item) preference matrix in Section 6.2. Furthermore, we present that as a memory-based CF algorithm, listwise CF inherently shares a similar prediction strategy under certain constraint to pointwise and pairwise CF algorithms in Section 6.3.

6.1. Pointwise CF

Let $U = \{u_1, u_2, \dots, u_M\}$ be the user set and $I = \{i_1, i_2, \dots, i_N\}$ be the item set, where the cardinalities $|U| = M$ and $|I| = N$. Pointwise collaborative filtering (CF), referred to as rating-oriented CF, recommends items to users based on the user-item rating matrix $R_{M \times N}$.

$$R = \begin{bmatrix} r_{u_1, i_1} & r_{u_1, i_2} & \cdots & r_{u_1, i_N} \\ r_{u_2, i_1} & r_{u_2, i_2} & \cdots & r_{u_2, i_N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{u_M, i_1} & r_{u_M, i_2} & \cdots & r_{u_M, i_N} \end{bmatrix}$$

THEOREM 6.1. *Let $R_{M \times N}$ be the rating matrix, and \bar{r}_u be the average rating of u . For the normalized rating matrix $R'_{M \times N}$, where each element $r'_{u,i} = r_{u,i} - \bar{r}_u$ is the normalized value of $r_{u,i}$, the similarity of the Pearson correlation coefficient between two users u and v is equal to their cosine similarity. Formally,*

$$\rho(\mathbf{u}', \mathbf{v}') = \cos(\mathbf{u}', \mathbf{v}'),$$

where \mathbf{u}' and \mathbf{v}' are the normalized vectors of the users u and v respectively.

Please refer to the proof of the theorems in Appendix, and the same below.

6.2. Pairwise CF

Similar to the user-item rating matrix in pointwise CF, in this section, we attempt to formulate the representation of the pairwise CF as the user-(item-item) pairwise preference matrix $P_{M \times N(N-1)}$.

DEFINITION 6.1 (USER-(ITEM-ITEM) PAIRWISE PREFERENCE MATRIX). *Given a rating matrix $R_{M \times N}$ of M users and N items, the user-(item-item) pairwise preference matrix $P_{M \times N(N-1)}$ is defined as follows:*

$$P = \begin{bmatrix} p_{u_1, (i_1, i_2)} & p_{u_1, (i_1, i_3)} & \cdots & p_{u_1, (i_N, i_{N-1})} \\ p_{u_2, (i_1, i_2)} & p_{u_2, (i_1, i_3)} & \cdots & p_{u_2, (i_N, i_{N-1})} \\ \vdots & \vdots & \ddots & \vdots \\ p_{u_M, (i_1, i_2)} & p_{u_M, (i_1, i_3)} & \cdots & p_{u_M, (i_N, i_{N-1})} \end{bmatrix},$$

where each element $p_{u, (i, j)} \in \{1, -1\}$ is the pairwise preference between items (i, j) for user u if u has rated both items i and j , which is defined in Equation (5). Otherwise, $p_{u, (i, j)}$ is unknown for prediction.

THEOREM 6.2. *Let $P_{M \times N(N-1)}$ be a user-(item-item) pairwise preference matrix. The Kendall's τ correlation coefficient between two users u and v based on rating matrix R is equal to their cosine similarity and Pearson correlation coefficient based on the matrix P . Formally,*

$$\tau(u, v) = \cos(u, v) = \rho(u, v).$$

THEOREM 6.3. *The prediction formula of pairwise CF based on the pairwise preferences is equivalent to the prediction formula of pointwise CF based on the representation of the user-(item-item) preference matrix.*

Both pairwise CF and pointwise CF are memory-based algorithms, which make predictions for each user based on a set of most similar users. Since the similarity calculation and prediction formulas between these two CF paradigms are equivalent according to Theorems 6.2 and 6.3, *pairwise CF actually performs a pointwise CF algorithm on a user-(item-item) preference matrix:*

$$\text{Pairwise CF} = \text{Pointwise CF} + \text{PairwisePreference Matrix}.$$

6.3. Listwise CF

In this section, we attempt to formulate the representation of listwise CF with a listwise preference matrix based on the *top-k probability model* introduced in Section 4.1, referred to as the top-k permutation probability matrix Θ_{top-k} in this paper. The definition of the top-k permutation probability matrix Θ_{top-k} is defined as follows:

DEFINITION 6.2 (TOP-k PERMUTATION PROBABILITY MATRIX). *Let $R_{M \times N}$ be a rating matrix with M users and N items. Let \mathcal{G}_k^I be the collection of the top-k permutation sets over items $I = \{i_1, i_2, \dots, i_N\}$ (see Definition 4.2). The top-k permutation probability matrix Θ_{top-k} is defined as follows:*

$$\Theta_{top-k} = \begin{bmatrix} \theta_{u_1, g_1} & \theta_{u_1, g_2} & \cdots & \theta_{u_1, g_{|\mathcal{G}_k^I|}} \\ \theta_{u_2, g_1} & \theta_{u_2, g_2} & \cdots & \theta_{u_2, g_{|\mathcal{G}_k^I|}} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{u_M, g_1} & \theta_{u_M, g_2} & \cdots & \theta_{u_M, g_{|\mathcal{G}_k^I|}} \end{bmatrix},$$

where the element $\theta_{u,g}$ is the probability of the top- k permutation set g for user u if u has rated all of the top- k items in g , which can be calculated according to Lemma 4.1. Otherwise, $\theta_{u,g}$ is unknown for prediction.

In the top- k permutation probability matrix Θ_{top-k} , only a few fraction of elements are observed. The element $\theta_{u,g}$ is unknown if the user u has not rated any of the top- k items in g . Thus the listwise paradigm of collaborative filtering can be defined as making predictions for the unobserved elements in Θ_{top-k} , based on which ranking lists can be aggregated and the ranking list with the highest probability is recommended for each user. Specially when $k = 1$, the element $\theta_{u,i}$ is observed if $i \in I_u$. Otherwise, for each item $j \in I/I_u$, $\theta_{u,j}$ is unknown for prediction.

Different from pointwise CF and pairwise CF, ListCF cannot directly use the elements of the matrix to calculate similarities and make predictions. When calculating similarity between users u and v , ListCF firstly filters the probabilities of the top- k permutation sets over their co-rated item set $I_{u,v}$ to form a new distribution, i.e., normalizing the filtered probabilities so that the summation is equal to 1. Then the similarity between users can be calculated with Equation (7). When predicting the rankings of the unrated items T_u for user u , ListCF cannot just predict the probability of the top- k permutation set $g \in \mathcal{G}_k^{T_u}$ as the weighted average top- k probabilities of his neighbors. Every element $\theta_{u,i}$ in Θ_{top-k} is a probability, which has been normalized by the summation of the top- k permutations over u 's rated items I_u . For example, the probability that the item i_1 is ranked highest among items $\{i_1, i_2\}$ is different from the probability that i_1 is ranked highest among $\{i_1, i_2, i_3\}$. Thus it is meaningless to average these normalized probabilities for prediction.

Despite these differences, as a memory-based CF algorithm, ListCF inherently shares a similar prediction strategy under certain constraint to pointwise and pairwise CF algorithms.

THEOREM 6.4. *ListCF predicts the probability of a top- k permutation set as the weighted average of the neighbors' probabilities of the corresponding top- k permutation sets, if $\forall v \in N_u : T_{u,v} \equiv T_u$ holds. Formally, for $\forall g \in \mathcal{G}_k^{T_u}$,*

$$\hat{P}_u(g) = \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\sum_{v \in N_u} s(u, v)}.$$

Theorem 6.4 presents the prediction formula of ListCF in a special case when all the neighbor users have rated the items in test set T_u of user u , i.e., $\forall v \in N_u : T_{u,v} \equiv T_u$ holds. In this case, for $\forall g \in \mathcal{G}_k^{T_u}$, the probability $\hat{P}_u(g)$ can be predicted as the weighted average of the neighbors' corresponding probabilities, which is the same as the pairwise preference prediction formula in pairwise memory-based CF [Liu and Yang 2008; Wang et al. 2012b; 2014] as well as the rating prediction formula in the pointwise memory-based CF after performing the normalization of the ratings, i.e., the average rating of the user u has been deducted from each rating $r_{u,i}$ of u , formally $r_{u,i}^{(N)} = r_{u,i} - \bar{r}_u$.

Now we explain why we cannot obtain such a result without the constraint of $\forall v \in N_u : T_{u,v} \equiv T_u$ in ListCF while we can with the constraint. In pointwise CF and pairwise CF, the ratings and the pairwise preferences of users are constant. However, in listwise CF, the probability distributions of each pair of users u and v in the ranking prediction phase vary with respect to the set of items $T_{u,v}$: (1) Given a target user u , for any two different neighborhood users v_1 and v_2 , since $T_{u,v_1} \neq T_{u,v_2}$ and $\mathcal{G}_k^{T_{u,v_1}} \neq \mathcal{G}_k^{T_{u,v_2}}$,

the length of the probability distribution \hat{P}_u of user u could change and the values in \hat{P}_u could be different; (2) Given two different target users u_1 and u_2 who have a same neighborhood user v , the probability distribution of P'_v for user v could also be different. Thus we cannot simply make predictions with a weighted average formula as pointwise and pairwise CF do. However, with the constraint of $\forall v \in N_u : T_{u,v} \equiv T_u$, $\mathcal{G}_k^{T_{u,v}} = \mathcal{G}_k^{T_u}$, the probabilities $\hat{P}_u(g)$ and $P'_v(g)$ ($\forall g \in \mathcal{G}_k^{T_u}$) in the ranking prediction formula are invariable, which is the same as those in pointwise CF and pairwise CF. In this case, we can obtain the same result shown in Theorem 6.4.

7. EXPERIMENTAL SETTINGS

7.1. Datasets

We conducted a series of experiments on three real-world rating datasets: Movielens-1M, EachMovie, and Netflix. All of these datasets are commonly used benchmark datasets in evaluating recommendation performance. The ratings in Movielens-1M and Netflix are on a scale from 1 to 5, while the EachMovie rating scale is from 1 to 6. In order to guarantee that there are adequate number of ratings per user for training and testing, we removed those users who have rated less than 20 items from these datasets. The statistics of the datasets are presented in Table I, where the sparsity levels in the last row are calculated as follows [Sarwar et al. 2001]:

$$Sparsity = 1 - \frac{\#rating\ scores}{\#users \times \#items}.$$

Table I. Statistics on the datasets.

	Movielens	EachMovie	Netflix
#users	6,040	36,656	429,584
#items	3,952	1,623	17,770
#ratings	1,000,209	2,580,222	99,884,940
#ratings/user	165.6	70.4	232.5
#ratings/item	253.1	1589.8	5621.0
sparsity	93.7%	95.7%	98.7%

7.2. Comparison Algorithms

In our experiments, we mainly compared the performance of ListCF with three memory-based algorithms of PointCF [Breese et al. 1998], EigenRank [Liu and Yang 2008] and VSRank [Wang et al. 2012b; 2014]. A direct comparison of them will provide valuable and irreplaceable insights. Besides, we also choose four model-based algorithms of CofiRank [Weimer et al. 2007], ListRank-MF [Shi et al. 2010], SVD-Feature [Chen et al. 2012] and Factorization machines (FM) [Rendle 2012] to further demonstrate the promising performance of ListCF. The detailed descriptions of their implementations are as follows:

- **PointCF** is a pointwise memory-based CF algorithm, which uses the Pearson correlation coefficient to calculate the similarities between users, and orders the items according to the predicted ratings for each user.
- **EigenRank** is a pairwise memory-based CF algorithm. It measures similarities between users with Kendall's τ correlation coefficient on users' pairwise preferences over their co-rated items. The final rankings of items are aggregated with a greedy method for recommendation.

- **VSRank** is pairwise memory-based CF algorithm, which represents each user's pairwise preferences based on the vector space model. It also uses a greedy method for discovery of an approximately optimal ranking.
- **CofiRank** is a listwise model-based CF algorithm that directly optimizes NDCG measure for recommendation. The implementation of CofiRank is based on the publicly available package online³. The dimensionality of latent features is set to be 10.
- **ListRank-MF** makes use of listwise learning to rank technique with matrix factorization (MF) to rank items. The implementation of ListRank-MF is publicly available online⁴. The dimensionality of latent features is 10 and $\lambda = 0.01$.
- **SVDFeature** is designed to efficiently solve large-scale collaborative filtering problems with auxiliary information. The implementation of SVDFeature is available online⁵. In SVDFeature, we set the dimensionality of latent features to be 10 and the regularization parameters of item factors and user factors to be 0.005. The learning rate and number of iterations for stochastic gradient descent is 0.01 and 200. The ranking of items is generated in the same way as PointCF.
- **FM** is a generic approach that allows to mimic most factorization models by feature engineering. The implementation of FM in our experiments is downloaded from the website⁶. In FM, we choose the stochastic gradient descent method for optimization. We set the dimensionality of latent features to be 10, and the learning rate to be 0.01.

7.3. Evaluation Metric

For rating-based CF algorithms, the standard evaluation criteria are the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE). Both criteria measure the error between true ratings and predicted ratings. Since our study focuses on improving item rankings instead of rating prediction, we employ the Normalized Discounted Cumulative Gain (NDCG) [Järvelin and Kekäläinen 2000; Järvelin and Kekäläinen 2002] as the evaluation metric, which is the most popular accuracy metric in information retrieval for evaluating ranking of retrieved documents with multi-level relevance scores.

In the context of collaborative filtering, item ratings assigned by users can naturally serve as relevance judgments. The NDCG metric is evaluated over some number of the top items on the ranked item list. Let U be the set of users and r_u^p be the rating score assigned by user u to the item at the p th position of the ranked list. The NDCG value at the n th position with respect to the given user u is defined as follows:

$$NDCG_u@n = Z_u \sum_{p=1}^n \frac{2^{r_u^p} - 1}{\log(1 + p)},$$

where Z_u is a normalization factor calculated so that the NDCG value of the optimal ranking is 1. $NDCG@n$ takes the mean of the $NDCG_u@n$ over the set of users U , which is defined as follows:

$$NDCG@n = \frac{1}{|U|} \sum_{u=1}^{|U|} NDCG_u@n,$$

where $|U|$ is the cardinality of the set of users.

³<http://www.cofirank.org/>

⁴<http://mmc.tudelft.nl/users/yue-shi>

⁵http://svdfeature.apexlab.org/wiki/Main_Page

⁶<http://www.libfm.org>

For the datasets of Movielens, EachMovie and Netflix, we randomly select each user's 10 rated items and their corresponding ratings to form test set, and the remaining ratings are used to form train set.

8. EXPERIMENTAL RESULTS

8.1. Values of k

For illustration purpose, we randomly selected 1000 users from the MovieLens-1M dataset and conducted experiments to compare the ranking accuracy as well as the runtime of the similarity calculation and ranking prediction phases (Phase I and II respectively) of ListCF when $k = 1, 2$ and 3. The size of neighborhood users was set to be 50, and 50 unrated items were predicted for each user in Phase II.

Table II. Accuracy and efficiency comparison of ListCF with different k .

	$k=1$	$k=2$	$k=3$
Accuracy (NDCG@5)	0.7866	0.7953	0.8052
Runtime of Phase I	8s	71s	3463s
Runtime of Phase II	1.5s	20.2s	731.7s

The results are reported in Table II. From the table we can see that, when $k > 1$ in ListCF, the improvement in recommendation accuracy is very subtle, but the degeneration of efficiency is extremely significant. For example, ListCF with $k = 3$ achieves about 0.02 improvement in NDCG@5 over ListCF with $k = 1$, but it takes 433 and 488 times longer in Phase (I) and Phase (II) respectively. Obviously, the loss outweighs the gain. Thus we set $k = 1$ when compare ListCF with other recommendation algorithms in the following experiments.

8.2. Efficiency

In this section, we experimentally compare our ListCF with the other two paradigms of memory-based collaborative filtering (CF) algorithms, PointCF and EigenRank, to demonstrate the efficiency gains of our algorithm.

We conducted experiments to compare their runtime of the similarity calculation phase and ranking prediction phase on the datasets. Experiments were run on a computer with 4 core 2 GHz CPU and 64 GB RAM. Due to the huge size of the Netflix data, we selected users from Netflix who had rated no more than 50 items to form a smaller dataset containing about 100,000 users. As analyzed in Section 4.5, when the number of items each user has rated increases the difference between EigenRank and ListCF would be more distinct, and we omit the result here.

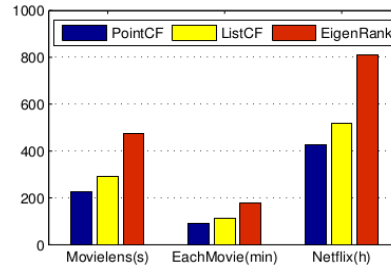


Fig. 1. Runtime comparison of the similarity calculation phase.

Figure 1 demonstrates the respective runtime of the similarity calculation phase of PointCF, ListCF and EigenRank on three datasets. Note that the units of the runtime are Second (s), Minute (min) and Hour (h) for the datasets of Movielens-1M, Each-Movie and Netflix respectively. From Figure 1, we can observe that in the similarity calculation phase:

- The runtime of ListCF is comparable with PointCF, where ListCF is merely around 1.2 times higher than PointCF. The slight increase of the runtime of ListCF mainly results from an extra step of computing the top-1 probability distributions for each pair of users.
- The runtime of ListCF is much lower than that of EigenRank, saving almost 40% time. This observation is consistent with our theoretical analysis in Section 4.5, which is that in the similarity calculation phase, the computational cost of pairwise CF is $(n + \bar{n}^2)/(n + \bar{n})$ higher than that of ListCF, where n is the average number of items rated by users, and \bar{n} is the average number of co-rated items by each pair of users.

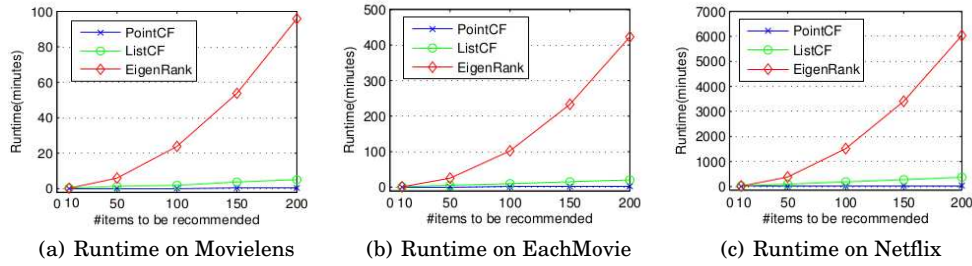


Fig. 2. Runtime comparison of the ranking prediction phase.

Figure 2 shows the runtime comparison of the ranking prediction phase of the three CF algorithms, where the horizontal axis of the figures is the number of items to be predicted for each user, and the vertical axis is the runtime of the algorithms. From the figures we can see that ListCF demonstrates significant improvement in prediction efficiency in comparison with EigenRank.

- From Figure 2(a), 2(b) and 2(c) we can see that the runtime of both ListCF and PointCF is linearly increased with the growth of the number of items to be predicted, while the increase of EigenRank (the pairwise CF) is quadratic. This observation is consistent with our theoretical analysis in Section 4.5 that, in the ranking prediction phase, the computational costs of ListCF and pairwise CF are $\mathcal{O}(M(N - n)ld)$ and $\mathcal{O}(M(N - n)^2l)$ respectively, where M is the number of users, and $N - n$ is the number of items to be predicted. l is the size of the neighbor set, and d is the maximal number of iterations in the gradient descent based optimization method.
- The runtime of ListCF is significantly reduced as compared to EigenRank even when no more than 200 items are predicted. For example, for the EachMovie dataset, ListCF takes 20min to predict the total rankings of 200 items for 36656 users, while the runtime of EigenRank is 423min, which is 21 times longer. The results on Movielens-1M and Netflix are 19 times and 16 times longer respectively.

Figure 3 shows the runtime ratio of EigenRank to ListCF in the ranking prediction phase, where the horizontal axis is the number of items to be predicted for each user, and the vertical axis is the runtime ratio of EigenRank to ListCF.

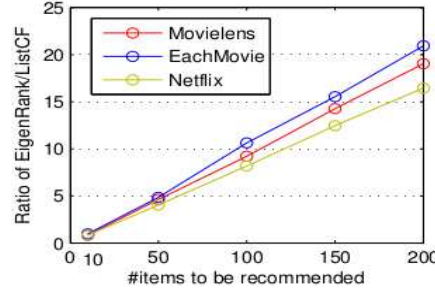


Fig. 3. Runtime ratio of EigenRank to ListCF in the ranking prediction phase.

From Figure 3 we can see that the runtime ratio of EigenRank to ListCF increases linearly with the growth of the number of items to be predicted in the ranking prediction phase. This observation is consistent with our analyzed result $(N - n)/d$ in Section 4.5, where $N - n$ is the number of items to be predicted, and d is a constant, i.e., the maximal number of iterations in the gradient descent optimization method. The gradients of the curves are slightly different because some iterative optimization processes could converge before reaching the maximal number of iterations d .

8.3. Accuracy

To demonstrate the recommendation performance of our model, we compare ListCF with all of the state-of-the-art recommendation algorithms mentioned in Section 7.2 on Movielens-1M, EachMovie and Netflix.

Figures 4 and 5 show the performance comparison results in terms of NDCG@1, 3 and 5 on the three datasets. From the figure we can observe that:

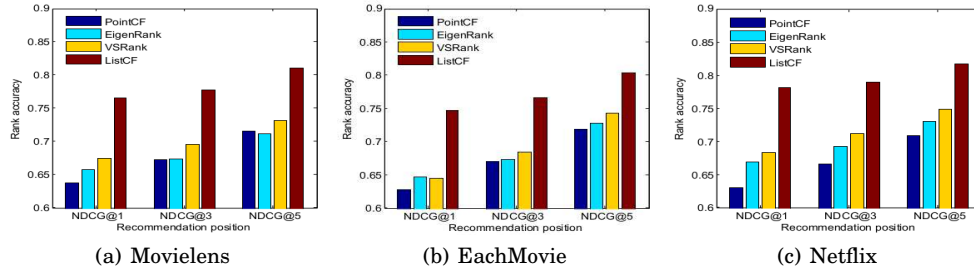


Fig. 4. Ranking performance comparisons with memory-based algorithms.

- ListCF sharply outperforms all of the memory-based algorithms in terms of NDCG@1, 3 and 5. The improvement of ListCF is more distinct when we consider higher positions in NDCG. For example, for Movielens-1M, ListCF improves 19.88%, 14.97%, and 13.41% in terms of NDCG@1, 3 and 5 in comparison with PointCF, and these numbers are 16.29%, 15.51%, and 13.93% respectively in comparison with EigenRank. For EachMovie and Netflix, we can get similar trends. This is because ListCF considers the ranking position issue in the similarity calculation phase, which is demonstrated in Example 4.4 in Section 4.5.
- ListCF can also significantly outperforms the state-of-the-art model-based algorithms CofiRank and ListRank-MF. For example, ListCF improves 11.19% over Cofi-

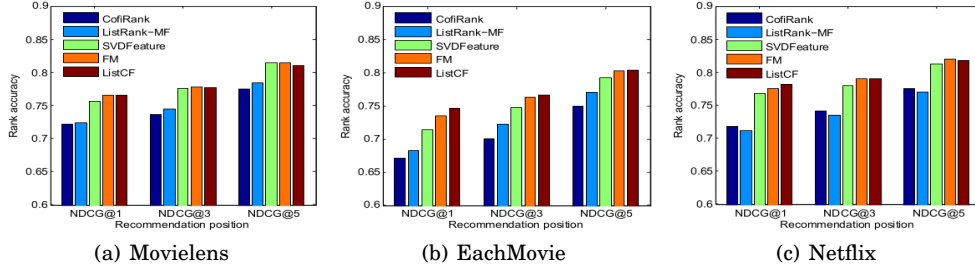


Fig. 5. Ranking performance comparisons with model-based algorithms.

Rank and 9.35% over ListRank-MF in terms of NDCG@1 on Movielens-1M, and these numbers are 8.84% and 9.86% respectively on Netflix.

- ListCF can outperform SVDFeature in most of conditions, though the improvement is subtle. For example, the improvement in terms of NDCG@1 and 3 is only 1.22% and 0.18% on Movielens, while these numbers are 1.87% and 1.32% on Netflix. Besides, ListCF can achieve comparable performance against FM. In comparison with FM, the highest accuracy gain of ListCF is 1.65% that is achieved on EachMovie in terms of NDCG@1. The other accuracy differences between these two algorithms are all less than 0.8%.
- We perform a two-tailed paired t-test to evaluate the significance of the accuracy gains against each baseline algorithm. In comparison with all of the memory-based algorithms and two model-based algorithms CofiRank and ListRank-MF, all of our improvements are strongly significant for $\alpha = 0.01$. Compared with SVDFeature, some of the results are weak significant for $\alpha = 0.05$, including improvement in terms of NDCG@1 and 3 on EachMovie, and NDCG@1 on Netflix. For FM, most of differences are not significant except the weak significance of the performance gain in term of NDCG@1 on EachMovie.

To sum up, ListCF can sharply outperform state-of-the-art memory-based CF algorithms in both efficiency and accuracy. Besides, As a memory-based CF algorithm, our proposed ListCF demonstrates superiority over two state-of-the-art model-based algorithms CofiRank and ListRank-MF, and achieves comparable performance to two other model-based algorithms SVDFeature and FM, although these model-based algorithms involve feature engineering and fine parameter tuning.

8.4. Impact of Parameters

Number of neighborhood users. In the similarity calculation phase, we choose a fixed number of most similar users to form the set of neighborhood users for each target user. Generally speaking, the number of neighborhood users could affect the performance of the memory-based CF algorithms. In this experiment, we used NDCG@5 as the evaluation measure investigate the performance of ListCF with respect to different numbers of neighborhood users.

Figure 6 shows the experimental results on three datasets when the neighborhood size increases from 10 to 500. From the figure we can observe that the NDCG@5 gradually increases as the neighborhood size increases from 10 to 100 because more similar users are selected to provide more reliable information for prediction. However, when the neighborhood size continues to increase, the performance of ListCF tends to be steady or even worse because many less similar neighbors are introduced, which have less or even downside effect on the predictions. In the following experiments, we set the neighborhood size to be 100 in Movielens-1M and EachMovie, and 200 in Netflix.

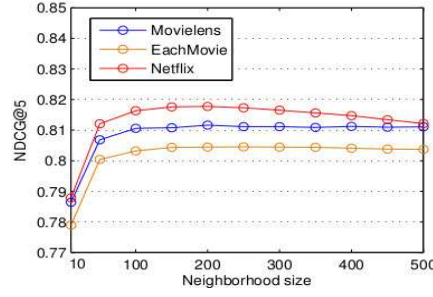


Fig. 6. Impact of neighborhood size on recommendation performance.

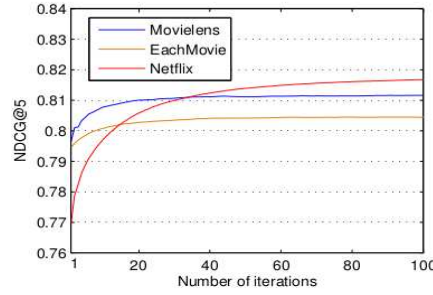


Fig. 7. Impact of iteration number on recommendation performance.

Number of iterations. In the ranking prediction phase of ListCF, we used a gradient decent method to update the variables iteratively. Here, we are interested in the convergence speed of ListCF. Figure 7 shows the changes of recommendation performance of ListCF with the number of iterations increasing on three datasets. From the figure, we can conclude that ListCF can reach a converged result within a very small number of iterations, i.e., 20 iterations in Movielens-1M and EachMovie, 50 iterations in Netflix on average.

8.5. Performance of Incremental ListCF

In this section, we experimentally evaluate the efficiency gains of the incremental ListCF. We randomly sampled 1000, 2000 and 5000 users and their ratings from Movielens-1M dataset to form three datasets.

Note that when implementing incremental ListCF, the old values of $D_{KL}(P_v \| P_u)$ and $D_{KL}(P_u \| P_v)$ are stored in the memory. To be more efficient, the values of $\sum_{j \in I_{u,v}} \phi(r_{u,j})$ and $\sum_{j \in I_{u,v}} \phi(r_{v,j})$ are also stored in advance, and hence we don't need to find the co-rated item set for each pair of users every time we update the user-user similarity.

Experiments have been carried out on a computer with 4 core 2 GHz CPU and 64 GB RAM. We ran each experiment 10 times and reported the average runtime. TableIII presents the runtime of the similarity recalculation phase in ListCF and incremental ListCF for Case 1 (i.e., IListCF-Case 1) and Case 2 (i.e., IListCF-Case 2) when a user updates a rating to the three datasets, where \bar{n} is the average number of co-rated items for each pair of users. From TableIII, we can observe that:

- The update time of incremental ListCF is much lower than ListCF. The time ratio of ListCF to incremental ListCF is approximately equals to \bar{n} . For example, for the

Table III. Runtime comparison of similarity updating in ListCF and incremental ListCF.

Datasets	\bar{n}	ListCF	IListCF-Case1	IListCF-Case2
1000 users	27	1.535s	0.057s	0.060s
2000 users	36	7.130s	0.201s	0.212s
5000 users	27	38.689s	1.409s	1.497s

dataset with 5000 users, the time ratio of ListCF to IListCF-Case1 is 27.46, which is quite close to the average number of co-rated items $\bar{n} = 27$.

- The update time of IListCF-Case2 is slightly higher than that of IListCF-Case1 because the similarity update formula for Case 2, Equation (15), is more complex than that for Case 1, Equation (14).
- The update time of ListCF and incremental ListCF are both quadratic to the number of users, which is inevitable since similarity between each pair of users needs to be updated.

9. CONCLUSION

In this paper, we propose ListCF, a listwise ranking-oriented collaborative filtering (CF) algorithm for recommender systems. ListCF utilizes the Plackett-Luce model to represent each user as a probability distribution of top- k permutations over rated items. Following the memory-based CF framework, ListCF measures similarity between users based on the Kullback-Leibler divergence between users' probability distributions over their commonly rated items. For each user, ListCF predicts the preference ranking over unrated items for recommendation based on the preferences of the most similar users. Besides, we provide the definitions of the user-(item-item) preference matrix for pairwise CF and the top- k permutation probability matrix for listwise CF, based on which we reveal insightful connections among these three paradigms of CF algorithms from the perspective of the matrix representations. To make ListCF more practical in real-world scenarios, we design incremental ListCF algorithm, which can incrementally update similarity between users when a user submits a new rating or updates an existing rating. Experimental results on three benchmark datasets show the promising performance of ListCF and incremental ListCF in both recommendation efficiency and accuracy in comparison with state-of-the-art CF algorithms.

In the future, first of all, we plan to explore other possible listwise representations and prediction loss functions to seek further improvement in recommendation accuracy. Second, it is very interesting to investigate model-based listwise CF algorithms based on the top- k permutation probability representations. Third but not last, we are going to integrate other data sources like users' comments and social connections to further improve the recommendation accuracy of our algorithms.

APPENDIX

In this appendix, we present the arithmetic derivations of the gradients in Section 4.3, the arithmetic derivations of the KL-Divergences in Section 5, and the proofs of the theorems in Section 6.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

This work was supported by the Academy of Finland (268078), the Natural Science Foundation of China (71402083, 61272240, 61502258), the Microsoft research fund (FY14-RES-THEME-25), the Doctoral Fund

of Ministry of Education of China (20110131110028), and the Natural Science Foundation of Shandong Province (ZR2014FQ007).

REFERENCES

- John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*. 43–52.
- Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*. 129–136.
- Edward Challis and David Barber. 2013. Gaussian Kullback-Leibler Approximate Inference. *Journal of Machine Learning Research* 14, 1 (2013), 2239–2286.
- Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: A Toolkit for Feature-based Collaborative Filtering. *The Journal of Machine Learning Research* 13, 1 (2012), 3619–3622.
- Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys)*. 39–46.
- Mukund Deshpande and George Karypis. 2004. Item-based Top-N Recommendation Algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177.
- David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using Collaborative Filtering to Weave An Information Tapestry. *Communications of ACM* 35, 12 (1992), 61–70.
- Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 230–237.
- Jonathan L. Herlocker, Joseph A. Konstan, and John T. Riedl. 2002. An Empirical Analysis of Design Choices in Neighborhood-based Collaborative Filtering Algorithms. *ACM Transactions on Information Systems* 5 (2002), 287–310. Issue 4.
- Thomas Hofmann. 2004. Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems* 22, 1 (2004), 89–115.
- Shanshan Huang, Jun Ma, Peizhe Cheng, and Shuaiqiang Wang. 2015a. A Hybrid Multigroup Cocustering Recommendation Framework Based on Information Fusion. *ACM Transactions on Intelligent Systems and Technology* 6, 2 (2015), 27:1–27:22.
- Shanshan Huang, Shuaiqiang Wang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. 2015b. Listwise Collaborative Filtering. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 343–352.
- Mohsen Jamali and Martin Ester. 2010. A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys)*. 135–142.
- Kalervo Järvelin and Jaana Kekäläinen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 41–48.
- Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.
- Meng Jiang, Peng Cui, Rui Liu, Qiang Yang, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2012. Social Contextual Recommendation. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM)*. 45–54.
- Minsuk Kahng, Sangkeun Lee, and Sang-goo Lee. 2011. Ranking in Context-aware Recommender Systems. In *Proceedings of the 20th International Conference Companion on World Wide Web (WWW)*. 65–66.
- Maurice G. Kendall. 1938. A New Measure of Rank Correlation. *Biometrika* 30, 1–2 (1938), 81–93.
- Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
- Yehuda Koren and Joe Sill. 2011. OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys)*. 117–124.
- Solomon Kullback. 1997. *Information Theory and Statistics*. Dover Publications.
- Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Proceedings of the 14th Conference on Neural Information Processing Systems (NIPS)*. 556–562.

- Xirong Li, Efstratios Gavves, Cees GM. Snoek, Marcel Worring, and Arnold WM. Smeulders. 2011. Personalizing Automated Image Annotation using Cross-Entropy. In *Proceedings of the 19th ACM Conference on Multimedia (MM)*. 233–242.
- Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com Recommendations: Item-to-item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.
- Zhenhua Ling and Lirong Dai. 2012. Minimum Kullback–Leibler Divergence Parameter Generation for HMM-Based Speech Synthesis. *IEEE Transactions on Audio, Speech and Language Processing* 20, 5 (2012), 1492–1502.
- Nathan N. Liu and Qiang Yang. 2008. Eigenrank: A Ranking-Oriented Approach to Collaborative Filtering. In *Proceedings of the 31st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 83–90.
- Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook*. Springer, 73–105.
- Xin Luo, Yunni Xia, and Qingsheng Zhu. 2012. Incremental Collaborative Filtering Recommender Based on Regularized Matrix Factorization. *Knowledge-Based Systems* 27 (2012), 271–280.
- John I. Marden. 1996. *Analyzing and Modeling Rank Data*. CRC Press.
- Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *Proceedings of Extended Abstracts of the 24th ACM Conference on Human Factors in Computing Systems (CHI)*. 1097–1101.
- Catarina Miranda and Alípio Mário Jorge. 2009. Item-Based and User-Based Incremental Collaborative Filtering for Web Recommendations. In *Proceedings of the 14th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence (EPIA)*. 673–684.
- Andriy Mnih and Ruslan Salakhutdinov. 2007. Probabilistic Matrix Factorization. In *Proceedings of the 21th Conference on Neural Information Processing Systems (NIPS)*. 1257–1264.
- Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, and Elias Theoharopoulos. 2005. Incremental Collaborative Filtering for Highly-scalable Recommendation Algorithms. In *Proceedings of the 15th International Conference on Foundations of Intelligent Systems (ISMIS)*. 553–561.
- Zhaochun Ren, Shangsong Liang, Edgar Meij, and Maarten de Rijke. 2013. Personalized Time-aware Tweets Summarization. In *Proceedings of the 36th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 513–522.
- Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), No. 57.
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Schmidt-Thie Lars. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*. 452–461.
- Jasson DM. Rennie and Nathan Srebro. 2005. Fast Maximum Margin Matrix Factorization for Collaborative Prediction. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*. 713–719.
- Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 5th ACM Conference on Computer Supported Cooperative Work (CSCW)*. 175–186.
- Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*. 285–295.
- Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2002. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. In *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT)*. 27–28.
- Abd-Krim Seghouane. 2011. A Kullback–Leibler Divergence Approach to Blind Image Restoration. *IEEE Transactions on Image Processing* 20, 7 (2011), 2078–2083.
- Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-based Recommender System. *Journal of Machine Learning Research* 6 (2005), 1265–1295.
- Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. In *Proceedings of the 6th ACM Conference on Recommender Systems (RecSys)*. 139–146.
- Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise Learning to Rank with Matrix Factorization for Collaborative Filtering. In *Proceedings of the 4th ACM conference on Recommender Systems (RecSys)*. 269–272.
- Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative Filtering Beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. *ACM Computer Surveys* 47, 1 (2014), 3:1–3:45.

- Luo Si and Rong Jin. 2003. Flexible Mixture Model for Collaborative Filtering. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*. 704–711.
- Herbert A. Simon. 1971. Designing organizations for an information rich world. In *Computers, Communications, and the Public Interest*. Baltimore, 37–72.
- Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1701–1708.
- Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. 2013. Exploiting Local and Global Social Context for Recommendation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*. 2712–2718.
- Slobodan Vucetic and Zoran Obradovic. 2005. Collaborative Filtering Using a Regression-Based Approach. *Knowledge and Information Systems* 7, 1 (2005), 1–22.
- Shuaiqiang Wang, Jiankai Sun, Byron J. Gao, and Jun Ma. 2012b. Adapting Vector Space Model to Ranking-based Collaborative Filtering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*. 1487–1491.
- Shuaiqiang Wang, Jiankai Sun, Byron J. Gao, and Jun Ma. 2014. VSRank: A Novel Framework for Ranking-based Collaborative Filtering. *ACM Transactions on Intelligent Systems and Technology* 5, 3 (2014), No.51.
- Yongchang Wang, Kai Liu, Qi Hao, Xianwang Wang, Daniel L. Lau, and Laurence G. Hassebrook. 2012a. Robust Active Stereo Vision using Kullback–Leibler Divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 3 (2012), 548–563.
- Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. 2007. CofiRank - Maximum Margin Matrix Factorization for Collaborative Ranking. In *Proceedings of the 21th Conference on Neural Information Processing Systems (NIPS)*. 1329–1336.
- Shuang-Hong Yang, Bo Long, Alexander J. Smola, Hongyuan Zha, and Zhaohui Zheng. 2011. Collaborative Competitive Filtering: Learning Recommender using Context of User Choice. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 295–304.
- Xiao Yang, Zhaoxin Zhang, and Ke Wang. 2012. Scalable Collaborative Filtering Using Incremental Update and Local Link Prediction. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM)*. 2371–2374.
- Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit Factor Models for Explainable Recommendation Based on Phrase-level Sentiment Analysis. In *Proceedings of the 37th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 83–92.

Online Appendix to: Ranking-oriented Collaborative Filtering: A Listwise Approach

SHUAIQIANG WANG, University of Jyväskylä
SHANSHAN HUANG, Shandong University
TIE-YAN LIU, Microsoft Research Asia
JUN MA, Shandong University
ZHUMIN CHEN, Shandong University
JARI VEIJALAINEN, University of Jyväskylä

A. ARITHMETIC DERIVATIONS OF THE GRADIENTS IN SECTION 4.3

Let

$$F(\varphi_u) = \sum_{v \in N_u} s(u, v) \cdot E(\hat{P}'_u, P'_v)$$

According to Equation (7) and Equation (9), the objective function F can be specified as follows:

$$\begin{aligned} F(\varphi_u) &= \sum_{v \in N_u} s(u, v) \cdot E(\hat{P}'_u, P'_v) \\ &= - \sum_{v \in N_u} s(u, v) \sum_{g \in \mathcal{G}_k^{T_{u,v}}} P'_v(g) \log_2(\hat{P}'_u(g)) \\ &= - \sum_{v \in N_u} s(u, v) \sum_{g \in \mathcal{G}_k^{T_{u,v}}} P'_v(g) \log_2 \left(\frac{\varphi_{u,g}}{\sum_{g' \in \mathcal{G}_k^{T_{u,v}}} \varphi_{u,g'}} \right) \\ &= \sum_{v \in N_u} s(u, v) \sum_{g \in \mathcal{G}_k^{T_{u,v}}} P'_v(g) \left[\log_2 \left(\sum_{g' \in \mathcal{G}_k^{T_{u,v}}} \varphi_{u,g'} \right) - \log_2(\varphi_{u,g}) \right]. \end{aligned}$$

The derivative of F with respect to $\varphi_{u,g}$ is:

$$\begin{aligned} \frac{\partial F}{\partial \varphi_{u,g}} &= \sum_{v \in N_u} \left(\frac{s(u, v) \sum_{g \in \mathcal{G}_k^{T_{u,v}}} P'_v(g)}{\ln 2 \cdot \sum_{g' \in \mathcal{G}_k^{T_{u,v}}} \varphi_{u,g'}} \right) - \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\ln 2 \cdot \varphi_{u,g}} \\ &= \sum_{v \in N_u} \frac{s(u, v)}{\ln 2 \cdot \sum_{g' \in \mathcal{G}_k^{T_{u,v}}} \varphi_{u,g'}} - \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\ln 2 \cdot \varphi_{u,g}}. \end{aligned}$$

B. ARITHMETIC DERIVATIONS OF THE KL-DIVERGENCES IN SECTION 5

B.1. Arithmetic Derivations in Section 5.1

Let $I_{u,v}$ be the set of items that commonly rated by users u and v before u rates i_a . After u submits the new rating r_{u,i_a} to i_a , the commonly rated set of items becomes $I'_{u,v} = I_{u,v} \cup \{i_a\}$. Let P'_u and P'_v be the probability distribution over the commonly rated set of items $I'_{u,v}$ for u and v respectively. The probability of the top-1 permutation with the top-1 item of $i \in I_{u,v}$ for user u is calculated as follows:

$$\begin{aligned} P'_u(i) &= \frac{\phi(r_{u,i})}{\sum_{j \in I'_{u,v}} \phi(r_{u,j})} = \frac{\phi(r_{u,i})}{\sum_{j \in I_{u,v}} \phi(r_{u,j}) + \phi(r_{u,i_a})} \\ &= \frac{\sum_{j \in I_{u,v}} \phi(r_{u,j})}{\sum_{j \in I_{u,v}} \phi(r_{u,j}) + \phi(r_{u,i_a})} \times \frac{\phi(r_{u,i})}{\sum_{j \in I_{u,v}} \phi(r_{u,j})} \end{aligned}$$

Thus

$$P'_u(i) = \begin{cases} \alpha P_u(i), & i \in I_{u,v} \\ 1 - \alpha, & i = i_a \end{cases}$$

where $\alpha = \frac{\sum_{j \in I_{u,v}} \phi(r_{u,j})}{\sum_{j \in I_{u,v}} \phi(r_{u,j}) + \phi(r_{u,i_a})}$. Similarly,

$$P'_v(i) = \begin{cases} \beta P_v(i), & i \in I_{u,v} \\ 1 - \beta, & i = i_a \end{cases}$$

where $\beta = \frac{\sum_{j \in I_{u,v}} \phi(r_{v,j})}{\sum_{j \in I_{u,v}} \phi(r_{v,j}) + \phi(r_{v,i_a})}$. Then, the KL-divergence of P'_u from P'_v can be calculated as follows:

$$\begin{aligned} D_{KL}(P'_v \| P'_u) &= \sum_{i \in I_{u,v}} P'_v(i) \log_2 \frac{P'_v(i)}{P'_u(i)} + P'_v(i_a) \log_2 \frac{P'_v(i_a)}{P'_u(i_a)} \\ &= \sum_{i \in I_{u,v}} \beta P_v(i) \log_2 \frac{\beta P_v(i)}{\alpha P_u(i)} + (1 - \beta) \log_2 \frac{1 - \beta}{1 - \alpha} \\ &= \sum_{i \in I_{u,v}} \beta P_v(i) \left(\log_2 \frac{\beta P_v(i)}{\alpha P_u(i)} + \log_2 \frac{\beta}{\alpha} \right) + (1 - \beta) \log_2 \frac{1 - \beta}{1 - \alpha} \\ &= \beta \sum_{i \in I_{u,v}} P_v(i) \log_2 \frac{P_v(i)}{P_u(i)} + \beta \log_2 \frac{\beta}{\alpha} \sum_{i \in I_{u,v}} P_v(i) + (1 - \beta) \log_2 \frac{1 - \beta}{1 - \alpha} \\ &= \beta D_{KL}(P_v \| P_u) + \beta \log_2 \frac{\beta}{\alpha} + (1 - \beta) \log_2 \frac{1 - \beta}{1 - \alpha} \end{aligned}$$

Both the calculation and the formula of the KL-divergence of P'_v from P'_u , $D_{KL}(P'_u \| P'_v)$, is very similar to $D_{KL}(P'_v \| P'_u)$, which is shown as follows:

$$\begin{aligned}
D_{KL}(P'_u \| P'_v) &= \sum_{i \in I_{u,v}} P'_u(i) \log_2 \frac{P'_u(i)}{P'_v(i)} + P'_u(i_a) \log_2 \frac{P'_u(i_a)}{P'_v(i_a)} \\
&= \sum_{i \in I_{u,v}} \alpha P_u(i) \log_2 \frac{\alpha P_u(i)}{\beta P_v(i)} + (1 - \alpha) \log_2 \frac{1 - \alpha}{1 - \beta} \\
&= \sum_{i \in I_{u,v}} \alpha P_u(i) \left(\log_2 \frac{\alpha P_u(i)}{\beta P_v(i)} + \log_2 \frac{\beta}{\alpha} \right) + (1 - \alpha) \log_2 \frac{1 - \alpha}{1 - \beta} \\
&= \alpha \sum_{i \in I_{u,v}} P_u(i) \log_2 \frac{P_u(i)}{P_v(i)} + \alpha \log_2 \frac{\alpha}{\beta} \sum_{i \in I_{u,v}} P_v(i) + (1 - \alpha) \log_2 \frac{1 - \alpha}{1 - \beta} \\
&= \alpha D_{KL}(P_u \| P_v) + \alpha \log_2 \frac{\alpha}{\beta} + (1 - \alpha) \log_2 \frac{1 - \alpha}{1 - \beta}.
\end{aligned}$$

B.2. Arithmetic Derivations in Section 5.2

Suppose user u updated his rating to item i_a from r_{u,i_a} to r'_{u,i_a} . The similarity between u and any other user v who has rated i_a ($i_a \in I_{u,v}$) needs to be recalculated. Since the commonly rated set of items $I_{u,v}$ and the ratings of v have not changed, the probability distribution P_v of user v stays the same. Let the probability distribution of user u changes from P_u to P'_u resulting from the update of the rating from r_{u,i_a} to r'_{u,i_a} by u . The probability of the top-1 permutation with the top-1 item of $i \in I_{u,v} \setminus \{i_a\}$ for user u is calculated as follows:

$$P'_u(i) = \frac{\phi(r_{u,i})}{\sum_{j \in I_{u,v} \setminus \{i_a\}} \phi(r_{u,j}) + \phi(r'_{u,i_a})} = \frac{\sum_{j \in I_{u,v}} \phi(r_{u,j})}{\sum_{j \in I_{u,v} \setminus \{i_a\}} \phi(r_{u,j}) + \phi(r'_{u,i_a})} \times \frac{\phi(r_{u,i})}{\sum_{j \in I_{u,v}} \phi(r_{u,j})}$$

Thus

$$P'_u(i) = \begin{cases} \gamma P_u(i), & i \in I_{u,v} \setminus \{i_a\} \\ \delta, & i = i_a \end{cases}$$

where $\gamma = \frac{\sum_{j \in I_{u,v}} \phi(r_{u,j})}{\sum_{j \in I_{u,v} \setminus \{i_a\}} \phi(r_{u,j}) + \phi(r'_{u,i_a})}$, and $\delta = \frac{\phi(r'_{u,i_a})}{\sum_{j \in I_{u,v} \setminus \{i_a\}} \phi(r_{u,j}) + \phi(r'_{u,i_a})}$. Thus, the KL-divergence of P'_u from P_v is as follows:

$$\begin{aligned}
&D_{KL}(P_v \| P'_u) \\
&= \sum_{i \in I_{u,v} \setminus \{i_a\}} P_v(i) \log_2 \frac{P_v(i)}{P'_u(i)} + P_v(i_a) \log_2 \frac{P_v(i_a)}{P'_u(i_a)} \\
&= \sum_{i \in I_{u,v} \setminus \{i_a\}} P_v(i) \log_2 \frac{P_v(i)}{\gamma P_u(i)} + P_v(i_a) \log_2 \frac{P_v(i_a)}{\delta} \\
&= \sum_{i \in I_{u,v}} P_v(i) \log_2 \frac{P_v(i)}{\gamma P_u(i)} - P_v(i_a) \log_2 \frac{P_v(i_a)}{\gamma P_u(i_a)} + P_v(i_a) \log_2 \frac{P_v(i_a)}{\delta} \\
&= \sum_{i \in I_{u,v}} P_v(i) \log_2 \frac{P_v(i)}{P_u(i)} + \left(\log_2 \frac{1}{\gamma} \right) \sum_{i \in I_{u,v}} P_v(i) - P_v(i_a) \log_2 \frac{P_v(i_a)}{\gamma P_u(i_a)} + P_v(i_a) \log_2 \frac{P_v(i_a)}{\delta} \\
&= D_{KL}(P_v \| P_u) - \log_2 \gamma + P_v(i_a) \log_2 \frac{\gamma P_u(i_a)}{\delta}
\end{aligned}$$

The KL-divergence of P_v from P'_u can be calculated as follows:

$$\begin{aligned}
& D_{KL}(P'_u \| P_v) \\
&= \sum_{i \in I_{u,v} \setminus \{i_a\}} P'_u(i) \log_2 \frac{P'_u(i)}{P_v(i)} + P'_u(i_a) \log_2 \frac{P'_u(i_a)}{P_v(i_a)} \\
&= \sum_{i \in I_{u,v} \setminus \{i_a\}} \gamma P_u(i) \log_2 \frac{\gamma P_u(i)}{P_v(i)} + \delta \log_2 \frac{\delta}{P_v(i_a)} \\
&= \sum_{i \in I_{u,v}} \gamma P_u(i) \log_2 \frac{\gamma P_u(i)}{P_v(i)} - \gamma P_u(i_a) \log_2 \frac{\gamma P_u(i_a)}{P_v(i_a)} + \delta \log_2 \frac{\delta}{P_v(i_a)} \\
&= \gamma \sum_{i \in I_{u,v}} P_u(i) \log_2 \frac{P_u(i)}{P_v(i)} + (\gamma \log_2 \gamma) \sum_{i \in I_{u,v}} P_u(i) - \gamma P_u(i_a) \log_2 \frac{\gamma P_u(i_a)}{P_v(i_a)} + \delta \log_2 \frac{\delta}{P_v(i_a)} \\
&= \gamma D_{KL}(P_u \| P_v) + \gamma \log_2 \gamma - \gamma P_u(i_a) \log_2 \frac{\gamma P_u(i_a)}{P_v(i_a)} + \delta \log_2 \frac{\delta}{P_v(i_a)}
\end{aligned}$$

C. PROOFS OF THE THEOREMS IN SECTION 6

THEOREM 6.1. *Let $R_{M \times N}$ be the rating matrix, and \bar{r}_u be the average rating of u . For the normalized rating matrix $R'_{M \times N}$, where each element $r'_{u,i} = r_{u,i} - \bar{r}_u$ is the normalized value of $r_{u,i}$, the similarity of the Pearson correlation coefficient between two users u and v is equal to their cosine similarity. Formally,*

$$\rho(\mathbf{u}', \mathbf{v}') = \cos(\mathbf{u}', \mathbf{v}'),$$

where \mathbf{u}' and \mathbf{v}' are the normalized vectors of the users u and v respectively.

PROOF. Since each element $r'_{u,i} = r_{u,i} - \bar{r}_u$ has been normalized, in the normalized rating matrix $R'_{M \times N}$, the new average rating $\bar{r}'_u = 0$ for $\forall u \in U$.

According to Equation (1), the Pearson correlation coefficient between two users u and v with the normalized matrix $R'_{M \times N}$ is:

$$\begin{aligned}
\rho(\mathbf{u}', \mathbf{v}') &= \frac{\sum_{i \in I_{u,v}} (r'_{u,i} - \bar{r}'_u) (r'_{v,i} - \bar{r}'_v)}{\sqrt{\sum_{i \in I_{u,v}} (r'_{u,i} - \bar{r}'_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r'_{v,i} - \bar{r}'_v)^2}} \\
&= \frac{\sum_{i \in I_{u,v}} r'_{u,i} \cdot r'_{v,i}}{\sqrt{\sum_{i \in I_{u,v}} (r'_{u,i})^2} \sqrt{\sum_{i \in I_{u,v}} (r'_{v,i})^2}} = \frac{\mathbf{u}'^\top \mathbf{v}'}{\|\mathbf{u}'\| \cdot \|\mathbf{v}'\|} = \cos(\mathbf{u}', \mathbf{v}'),
\end{aligned}$$

□

THEOREM 6.2. *Let $P_{M \times N(N-1)}$ be a user-(item-item) pairwise preference matrix. The Kendall's τ correlation coefficient between two users u and v based on rating matrix R is equal to their cosine similarity and Pearson correlation coefficient based on the matrix P . Formally,*

$$\tau(u, v) = \cos(u, v) = \rho(u, v).$$

PROOF. Let $I_{u,v}$ be the set of the items commonly rated by both users u and v , and the cardinality $|I_{u,v}| = n$. Based on the matrix $P_{M \times N(N-1)}$, the cosine similarity

between users u and v is:

$$\cos(u, v) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} = \frac{\sum_{i,j \in I_{u,v}} p_{u,(i,j)} p_{v,(i,j)}}{\sqrt{\sum_{i,j \in I_{u,v}} p_{u,(i,j)}^2} \sqrt{\sum_{i,j \in I_{u,v}} p_{v,(i,j)}^2}},$$

where \mathbf{u} and \mathbf{v} are the pairwise preference vectors containing the pairwise preferences of user u and v between items (i, j) and $i, j \in I_{u,v}$.

According to the definition of $p_{u,(i,j)}$, $\sum_{i,j \in I_{u,v}} p_{u,(i,j)} p_{v,(i,j)} = 2(N_c - N_d)$ holds, where N_c and N_d are the numbers of the concordant pairs and discordant pairs according to the ratings of user u and v . Since $p_{u,(i,j)}^2 = p_{v,(i,j)}^2 = 1$ and $|I_{u,v}| = n$, $\sum_{i,j \in I_{u,v}} p_{u,(i,j)}^2 = \sum_{i,j \in I_{u,v}} p_{v,(i,j)}^2 = n(n-1)$. Thus

$$\cos(u, v) = \frac{2(N_c - N_d)}{n(n-1)} = \tau(u, v).$$

For each user, the mean value of the pairwise preferences is 0. According to Theorem 6.1, $\cos(u, v) = \rho(u, v)$ holds. Thus $\tau(u, v) = \cos(u, v) = \rho(u, v)$. \square

THEOREM 6.3. *The prediction formula of pairwise CF based on the pairwise preferences is equivalent to the prediction formula of pointwise CF based on the representation of the user-(item-item) preference matrix.*

PROOF. For each user u , the mean value of the pairwise preferences based on the representation of the user-(item-item) preference matrix equals zero, i.e., $\bar{p}_u = 0$. Thus the prediction formula of pairwise CF (Equation (4)) can be rewritten as follows

$$\Psi_u(i, j) = \frac{\sum_{v \in N_u^{i,j}} \tau(u, v) p_{v,(i,j)}}{\sum_{v \in N_u^{i,j}} \tau(u, v)} = \bar{p}_u + \frac{\sum_{v \in N_u^{i,j}} \tau(u, v) (p_{v,(i,j)} - \bar{p}_v)}{\sum_{v \in N_u^{i,j}} \tau(u, v)},$$

which is the same as the prediction formula of pointwise CF based on the representation of the user-(item-item) preference matrix (Equation (2)). \square

THEOREM 6.4. *ListCF predicts the probability of a top- k permutation set as the weighted average of the neighbors' probabilities of the corresponding top- k permutation sets, if $\forall v \in N_u : T_{u,v} \equiv T_u$ holds. Formally, for $\forall g \in \mathcal{G}_k^{T_u}$,*

$$\hat{P}_u(g) = \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\sum_{v \in N_u} s(u, v)}.$$

PROOF. If $\forall v \in N_u : T_{u,v} \equiv T_u$, then $\forall v \in N_u : G_k^{T_{u,v}} \equiv G_k^{T_u}$. For the optimization problem in Equation (10), set the derivative of F (Equation 12) to be 0:

$$\begin{aligned}
& \sum_{v \in N_u} \frac{s(u, v)}{\ln 2 \cdot \sum_{g' \in \mathcal{G}_k^{T_u, v}} \varphi_{u, g'}} - \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\ln 2 \cdot \varphi_{u, g}} \\
&= \frac{\sum_{v \in N_u} s(u, v)}{\ln 2 \cdot \sum_{g' \in \mathcal{G}_k^{T_u}} \varphi_{u, g'}} - \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\ln 2 \cdot \varphi_{u, g}} = 0.
\end{aligned}$$

Thus

$$\frac{\sum_{v \in N_u} s(u, v)}{\sum_{g' \in \mathcal{G}_k^{T_u}} \varphi_{u, g'}} = \frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\varphi_{u, g}},$$

and

$$\frac{\sum_{v \in N_u} s(u, v) P'_v(g)}{\sum_{v \in N_u} s(u, v)} = \frac{\varphi_{u, g}}{\sum_{g' \in \mathcal{G}_k^{T_u}} \varphi_{u, g'}} = \hat{P}_u(g).$$

□