Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Discrete Listwise Collaborative Filtering for Fast Recommendation

Chenghao Liu* Tao Lu† Zhiyong Cheng‡ Xin Wang§ Jianling Sun† Steven Hoi *

Abstract

Listwise collaborative filtering, which directly predicts a ranking list of items for the given user, achieves superior accuracy performance since it is aligned with the ultimate goals of recommender systems. However, in corpus with the enormous number of items, the calculation cost for the learnt model to predict all user-item preferences is tremendous, which makes full corpus retrieval extremely difficult. In this paper, we propose a binarized collaborative filtering method, called Discrete Listwise Collaborative Filtering (DLCF), to represent users and items as binary codes for fast recommendation. As such, the proposed method could accelerate the retrieval procedure, since the user-item similarity could be efficiently computed via Hamming distance. We further adopt the discrete coordinate descent method to jointly optimize our proposed model. Extensive experiments performed on three real-world datasets demonstrate that 1) DLCF significantly outperforms the state-of-the-art binarized recommendation methods, and 2) DLCF shows very competitive ranking accuracy compared to its real-valued version while significantly improving the retrieval efficiency.

1 Introduction

Top-k recommendation is a fundamental task of a recommender system, which aims to retrieve a set of preferred items for each user request from the entire corpus. However, this problem is quite challenging both from an accuracy and efficiency perspective [Kang and McAuley, 2019]. The enormous number of items makes the prediction problem difficult in terms of their variability and sparsity. And besides preciseness, recommender systems should strike a balance between accuracy and efficiency [Zhu *et al.*, 2019]. Practically, in corpus with tens or hundreds of millions of items, exhaustively searching over all items to calculate each item's preference score and then ranking them according to the scores for each user request is intractable due to its high latency.

In the literature of recommender systems, lots of research work focus on improving recommendation accuracy. Among these methods, Listwise Collaborative Filtering (LCF) [Cao *et al.*, 2007; Wang *et al.*, 2016] has demonstrated superior performance to others. To align with the ultimate goal of recommender systems, LCF models the ranking probability distributions over the permutations of items for each user. Compared to the conventional Collaborative Filtering (CF) [Koren *et al.*, 2009] method which predicts the preference of a user towards an item, LCF directly predicts the ranking list of items for the given user.

To solve the efficiency problem in recommender systems, Binarized Collaborative Filtering (BCF) techniques, which represent users/items with binary codes instead of the real-value vectors, were proposed to accelerate the retrieval process [Zhou and Zha, 2012; Zhang *et al.*, 2014, 2016]. In this way, the preference scores can be efficiently calculated by fast bit-operations in the Hamming space. Meanwhile, by exploiting special data structures for indexing all items, the computation complexity of generating top-k preferred items could be sub-linear or even constant [Wang *et al.*, 2012; Muja and Lowe, 2009]. In addition, each dimension of binary codes is only stored by one bit instead of $32/64$ bits float number for real-valued vectors, which remarkably reduces the storage cost.

In order to take advantage of the strengths of both the ranking accuracy of LCF and the retrieval efficiency of BCF, we propose a Discrete Listwise Collaborative Filtering method (DLCF) that learns binary codes for listwise CF model. To the best of our knowledge, this is the first work to perform hash technique on listwise loss for recommendation. Specifically, the challenges and our contributions are briefly summarized as:

1. The first challenge is how to effectively learn binary codes with the real-valued observed ratings. Instead of binarizing the learned user/item vectors from the LCF by thresholding, we directly optimize the binary codes with listwise loss which can preserve the intrinsic user-item relationship. By additionally imposing balanced and decorrelated constraints, DLCF is able to derive informative yet compact binary codes in a limited size.

2. The second challenge is how to efficiently optimize the listwise loss with balanced and decorrelated constraints. Due to the form of the sum of exponentials in listwise

*Salesforce Research Asia. {chenghao.liu, shoi}@salesfoce.com

†School of Computer Science and Technology, Zhejiang University. {lutaott,sunjl}@zju.edu.cn

‡Shandong Artificial Intelligence Institute, Qilu University of Technology (Shandong Academy of Sciences). jason.zy.cheng@gmail.com

§Department of Computer Science and Technology, Tsinghua University. xin_wang@tsinghua.edu.cn

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

loss, it is difficult to derive closed-form solutions for updating user/item binary codes, so that we seek its local quadratic upper bound. We then adopt an efficient alternative optimization algorithm by solving several mixed-integer programming subproblems in an iterative process. This optimization technique could preserve the intrinsic user-item similarity with the goal of top-k ranking.

3. Extensive experiments on real-world datasets demonstrate that DLCF significantly outperforms the state-of-the-art BCF methods. Compared with the real-valued LCF method, DLCF significantly improves retrieval efficiency and ensure competitive ranking accuracy.

2 Related Work

In this section, we briefly review related work on listwise collaborative filtering and binarized collaborative filtering.

2.1 Listwise Collaborative Filtering (LCF) Listwise CF [Cao *et al.*, 2007] aims to directly predict a ranking list of items for the given user. In this way, both the ranking positions and equal ratings can be well considered. It usually utilizes a permutation probability model to represent each user as a probability distribution over the permutations of rated items. Therefore, more "correct" permutations, where items with larger ranking scores are ranked higher, are assigned with larger probabilities. [Wang *et al.*, 2016] proposed to optimize the cross entropy between the probability of top-k items and the observed ranking, since the time complexity is exponential to k. But the cross-entropy loss may rank worse scoring permutations more highly, since rating data usually contain many ties. Instead, [Xia *et al.*, 2008] considered a maximum likelihood framework to use the permutation probability directly. Recently, [Wu *et al.*, 2018] extended this idea and proposed a stochastic queuing process for handling ties and missing data.

2.2 Binarized Collaborative Filtering (BCF) As a pioneer work, Locality-Sensitive Hashing has been adopted for generating binary codes for Google News readers [Das *et al.*, 2007]. Based on this work, rotating [Zhou and Zha, 2012] features to obtain binary codes was applied for mapping learned user/item embeddings into the binary codes for recommendation. In order to derive more compact binary codes, the de-correlated constraint over different binary codes was imposed on user/item continuous representations [Liu *et al.*, 2014b]. The relevant work could be summarized as two independent stages: relaxed learning of user/item representations with some specific constraints and subsequent bianary quantization. However, such two-stage approaches suffer from a large quantization loss according to [Zhang *et al.*, 2016], so direct optimization with discrete constraints was proposed. To derive compact yet informative binary codes, the balanced and de-correlated constraints were further imposed [Zhang *et al.*, 2016]. To handle social information from users, discrete social recommendation was proposed [Liu *et al.*, 2019b]. In order to incorporate content information from users and items, content-aware matrix factorization and factorization machine with binary constraints were also proposed [Lian *et al.*, 2017; Liu *et al.*, 2018]. In order to deal with ranking-oriented CF model, AUC objective function with pairwise loss was developed [Zhang *et al.*, 2017] to address personalized ranking from implicit feedback. Recently, self-paced learning has been used for optimizing the discrete and pairwise ranking-based objective [Zhang *et al.*, 2018]. To enhance the representation capability, composistional coding collaborative filtering [Liu *et al.*, 2019a] used both real-valued vectors and binary coding to strike a blance between retrieval efficiency and accuracy.

3 Discrete Listwise Collobiartive Filtering

3.1 Listwise Objective Function LCF utilizes Plackett-Luce model [Cao *et al.*, 2007; Marden, 2014; Wu *et al.*, 2018], a widely used permutation probability model, to represent each user as a probability distribution over the permutations of rated items. It learns two latent matrices, i.e., $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]^\top \in \mathbb{R}^{m\times r}$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{u}_n]^\top \in \mathbb{R}^{n\times r}$, where m and n denote the number of users and items, respectively. The score of user i towards j is the dot product of their latent vectors, which is formulated as $\mathbf{u}_i^\top \mathbf{v}^j$. Then, the permutation probability is a generative model for the ranking parametrized by user-item scores.

Formally, let Π_i be a particular permutation of the n items for user i, which is a random variable and takes values from the set of all possible permutations. Π_{i1} denotes the index of highest ranked item and Π_{in} is the lowest ranked. Then, the probability of generating the ranking permutation matrix $\Pi = [\Pi_1, \ldots, \Pi_m]^\top \in \mathbb{R}^{m\times n}$ for each user can be formulated as

$$P(\Pi|\mathbf{U}, \mathbf{V}) = \prod_{i=1}^{m} P_i(\Pi_i|\mathbf{u}_i, \mathbf{V}) = \prod_{i=1}^{m}\prod_{j=1}^{n} \frac{e^{\phi(\mathbf{u}_i^\top \mathbf{v}_{\Pi_{ij}})}}{\sum_{l=j}^{n} e^{\phi(\mathbf{u}_i^\top \mathbf{v}_{\Pi_{il}})}},$$

where $\phi(\cdot)$ is an increasing and strictly positive function. Obviously, larger ratings will tend to be ranked more highly than lower ratings. In recommender systems, we usually observe only a proportion of the ratings data R and it include many items with the same ratings for each user, which makes the permutation Π no longer unique and there is a set of permutations that coincides with rating. To address this limitation, we adopt the stochastic queuing process [Wu *et al.*, 2018] that shuffles the ordering of observed items with the same ratings. Specifically, we denote the set of valid permutations as $S(\Omega, R)$, where Ω is the set of all pairs (i, j) such that R_{ij}, the rating of user i for item j, is observed.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Then, the probability of generating the observed ratings R can be formulated as

$$\begin{aligned} P(R|\mathbf{U},\mathbf{V}) &= \sum_{\Pi\in S(R,\Omega)} P(\Pi|\mathbf{U},\mathbf{V}) \\ &= \sum_{\Pi\in S(R,\Omega)} \prod_{i=1}^{m} P_i(\Pi_i,\mathbf{u}_i,\mathbf{D}) \\ &= \sum_{\Pi\in S(R,\Omega)} \prod_{i=1}^{m}\prod_{j=1}^{n_i} \frac{e^{\phi(\mathbf{u}_i^\top \mathbf{v}_{\Pi_{ij}})}}{\sum_{l=j}^{n_i} e^{\phi(\mathbf{u}_i^\top \mathbf{v}_{\Pi_{il}})}}, \end{aligned} \tag{3.1}$$

where n_i denotes the number of observed items for user i. To learn $\mathbf{U}$ and $\mathbf{V}$, we can minimize the negative likelihood of permutation probability over observed ratings:

$$\underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} -\log \sum_{\Pi\in S(R,\Omega)} P(\Pi|\mathbf{U},\mathbf{V}). \tag{3.2}$$

3.2 Learning Model To improve recommendation efficiency, we represent users/items with binary codes instead of real-valued vectors such that the computation of dot product between user and item binary codes can be accelerated through the Hamming distance. Denote $\mathbf{B} = [\mathbf{b}_1,\ldots,\mathbf{b}_m]^\top \in \{\pm 1\}^{m\times r}$ and $\mathbf{D} = [\mathbf{d}_1,\ldots,\mathbf{d}_n]^\top \in \{\pm 1\}^{n\times r}$ respectively as r-length binary codes for m users and n items, then the score for user i and item j can be calculated by

$$\hat{r}_{ij} = \frac{1}{r}\sum_{k=1}^{r} \mathbb{I}(b_{ik} = d_{jk}) = 1 - \frac{1}{r}H(\mathbf{b}_i,\mathbf{d}_j) = \frac{1}{2} + \frac{1}{r}\mathbf{b}_i^\top \mathbf{d}_j$$

where $H(\cdot,\cdot) = \sum_{k=1}^{r}\mathbb{I}(b_{ik} = d_{jk})$ is the Hamming distance between two binary codes and $\mathbb{I}(\cdot)$ denotes the indicator function that returns 1 if the element is true and 0 otherwise. According to [Muja and Lowe, 2009; Wang *et al.*, 2012], the time complexity of dot product could be logarithmic or even constant through the high efficiency of bit operations.

In order to maximize the information each bit carries and to assure the compactness of the learned binary codes, we impose the de-correlated constraint on the binary codes to guarantee the independence among different bits and place the balance constraint on the learning process to ensure that each bit carries as much information as possible. Therefore, learning binary codes for users and items in listwise CF model is to minimize the following objective function:

$$\begin{aligned} &\underset{\mathbf{B},\mathbf{D}}{\operatorname{argmin}} -\log \sum_{\Pi\in S(R,\Omega)} P(\Pi|\mathbf{B},\mathbf{D}) \\ &\text{s.t.}\quad \underbrace{\mathbf{1}_m^\top\mathbf{B} = 0, \mathbf{1}_n^\top\mathbf{D} = 0}_{\text{balanced constraint}}, \underbrace{\mathbf{B}^\top\mathbf{B} = m\mathbf{I}_r, \mathbf{D}^\top\mathbf{D} = n\mathbf{I}_r}_{\text{de-correlated constraint}} \\ &\mathbf{B}\in\{\pm 1\}^{m\times r},\ \mathbf{D}\in\{\pm 1\}^{n\times r}, \end{aligned} \tag{3.3}$$

where we assume that $\phi(\mathbf{b}_i^\top\mathbf{d}_j) = \frac{\mathbf{b}_i^\top\mathbf{d}_j + r}{2r}$. As such, the value of $\phi(\mathbf{b}_i^\top\mathbf{d}_j)$ can be bounded in $[0,1]$. Due to the form of the logarithm of the sum of the exponentials in $P(\Pi|\mathbf{B},\mathbf{D})$ in (3.1) and (3.2), it is difficult to directly derive the updating rule for binary codes $\mathbf{B}$ and $\mathbf{D}$ even based on discrete coordinate descent method. However, the Log-Sum-Exp function $\log\sum_i e^{x_i}$ is convex with respect to x_i, so we seek its upper variational yet quadratic bound for efficient optimization:

$$\begin{aligned} &-\log \sum_{\Pi\in S(R,\Omega)} P(\Pi|\mathbf{B},\mathbf{D}) \le -\sum_{\Pi\in S(R,\Omega)} \log P(\Pi|\mathbf{B},\mathbf{D}) \\ &= \sum_{\Pi\in S(R,\Omega)}\sum_{i=1}^{m}\sum_{j=1}^{n_i}(\log\sum_{l=j}^{n_i} e^{\phi(\mathbf{b}_i^\top\mathbf{d}_{\Pi_{il}})} - \phi(\mathbf{b}_i^\top\mathbf{d}_{\Pi_{ij}})) \\ &\le \sum_{\Pi\in S(R,\Omega)}\sum_{i=1}^{m}\sum_{j=1}^{n_i}\omega_{ij}, \end{aligned} \tag{3.4}$$

where $\omega_{ij} = \alpha_{ij} + \sum_{l=j}^{n_i}\Big(\frac{\phi(\mathbf{b}_i^\top\mathbf{d}_{\Pi_{il}}) - \alpha_{ij} - \xi_{ijl}}{2} + \lambda(\xi_{ijl})((\phi(\mathbf{b}_i^\top\mathbf{d}_{\Pi_{il}}) - \alpha_{ij})^2 - \xi_{ijl}^2) + \log(1+e^{\xi_{ijl}})\Big) - \phi(\mathbf{b}_i^\top\mathbf{d}_{\Pi_{ij}})$ and $\lambda(\xi) = \frac{1}{2\xi}(\frac{1}{1+e^{-\xi}} - \frac{1}{2})$. $l = rank_i(j)$ is the rank of item j for user i. α_{ij} and ξ_{ijl} are variational parameters. The first inequality follows the Jensen's inequality and convexity of $-log$ function and the second inequality follows from the Bouchard bound in [Bouchard, 2007]. After introducing varitional parameters ξ and α, the objective function in (3.3) is re-formulated as:

$$\begin{aligned} &\underset{\mathbf{B},\mathbf{D},\alpha,\xi}{\operatorname{argmin}} \sum_{\Pi\in S(R,\Omega)}\sum_{i=1}^{m}\sum_{j=1}^{n_i}\omega_{ij} \\ &\text{s.t.}\quad \mathbf{1}_m^\top\mathbf{B} = 0, \mathbf{1}_n^\top\mathbf{D} = 0, \mathbf{B}^\top\mathbf{B} = m\mathbf{I}_r, \mathbf{D}^\top\mathbf{D} = n\mathbf{I}_r \\ &\mathbf{B}\in\{\pm 1\}^{m\times r},\ \mathbf{D}\in\{\pm 1\}^{n\times r}, \end{aligned} \tag{3.5}$$

where $\alpha = \{\alpha_{11},\ldots,\alpha_{mn_m}\}$ and $\xi = \{\xi_{111},\ldots,\xi_{mn_mn_m}\}$. The problem formulated in (3.5) is essentially a challenging discrete optimization problem, since it is generally NP-hard. Finding the global optimum solution needs to involve $O(2^{(rn+rm)})$ combinatorial search for the binary codes. An alternative way is to impose auxiliary continuous variables $\mathbf{X}\in\mathcal{B}$ and $\mathbf{Y}\in\mathcal{D}$, where $\mathcal{B} = \{\mathbf{X}\in\mathcal{R}^{m\times r}|\mathbf{1}_m\mathbf{X} = 0, \mathbf{X}^\top\mathbf{X} = m\mathbf{I}_r\}$ and $\mathcal{D} = \{\mathbf{Y}\in\mathcal{R}^{n\times r}|\mathbf{1}_n\mathbf{Y} = 0, \mathbf{Y}^\top\mathbf{Y} = n\mathbf{I}_r\}$. Then the balanced and de-correlated constraints can be softened by $d(\mathbf{B},\mathcal{B}) = \min_{\mathbf{X}\in\mathcal{B}}\|\mathbf{B}-\mathbf{X}\|_F$ and $d(\mathbf{D},\mathcal{D}) = \min_{\mathbf{Y}\in\mathcal{D}}\|\mathbf{D}-\mathbf{Y}\|_F$, respectively. Finally, we can solve problem (3.5) in a computationally tractable

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

manner:

$$\underset{\mathbf{B},\mathbf{D},\alpha,\xi,\mathbf{X},\mathbf{Y}}{\operatorname{argmin}} \sum_{\Pi\in S(R,\Omega)} \sum_{i=1}^{m}\sum_{j=1}^{n_i} \omega_{ij} + \beta_1 d^2(\mathbf{B},\mathcal{B}) + \beta_2 d^2(\mathbf{D},\mathcal{D})$$

$$\text{s.t.} \quad \mathbf{B}\in\{\pm 1\}^{m\times r},\ \mathbf{D}\in\{\pm 1\}^{n\times r}. \tag{3.6}$$

where β_1 and β_2 are hyperparameters. We can find that large values of β_1 and β_2 will force $d^2(\mathbf{B},\mathcal{B}) = d^2(\mathbf{D},\mathcal{D}) = 0$ when the objective in (3.6) can be optimized under some feasible constraints. On the other side, the comparative small values of β_1 and β_2 provide a certain discrepancy between $\mathbf{B}$ and $\mathbf{X}$, $\mathbf{D}$ and $\mathbf{Y}$, respectively. Note that the norm of binary codes are constants and do not take any effect on regularization terms. Therefore, we can replace the original regularization $d^2(\mathbf{B},\mathcal{B}), d^2(\mathbf{D},\mathcal{D})$ with $tr(\mathbf{B}^\top\mathbf{X}), tr(\mathbf{D}^\top\mathbf{Y})$ as follows:

$$\begin{aligned}\underset{\mathbf{B},\mathbf{D},\alpha,\xi\mathbf{X},\mathbf{Y}}{\operatorname{argmin}} & \sum_{\Pi\in S(R,\Omega)} \sum_{i=1}^{m}\sum_{j=1}^{n_i} \omega_{ij} \\ & - 2\beta_1 tr(\mathbf{B}^\top\mathbf{X}) - 2\beta_2 tr(\mathbf{D}^\top\mathbf{Y}) \\ \text{s.t.} \quad & \mathbf{1}_m^\top\mathbf{X} = 0, \mathbf{X}^\top\mathbf{X} = m\mathbf{I}_r, \mathbf{1}_n^\top\mathbf{Y} = 0, \mathbf{Y}^\top\mathbf{Y} = n\mathbf{I}_r \\ & \mathbf{B}\in\{\pm 1\}^{m\times r},\ \mathbf{D}\in\{\pm 1\}^{n\times r},\end{aligned} \tag{3.7}$$

where $tr(\cdot)$ denotes the trace of a matrix. It is worth noting that we do not discard the binary constraints but directly optimize the binary codes. Through joint optimization for the binary codes, variational parameters and the auxiliary real-valued variables, we can achieve nearly balanced and un-correlated binary codes.

3.3 Alternating Optimization We can employ alternating optimization strategy to solve the problem in (3.7). At each step, we randomly choose a permutation matrix $\Pi \in S(R,\Omega)$ as the stochastic queuing process in [Wu *et al.*, 2018] and then update $\mathbf{B}$, $\mathbf{D}$ ξ, α, $\mathbf{X}$, Y alternatively. The detailed algorithm description is presented in Algorithm 1. Below, we show that each subproblem has a closed-form solution.

Learning ξ and α: By fixing $\alpha, \mathbf{B}, \mathbf{D}$, the sub-objective w.r.t. ξ is convex since its second derivative is greater than zero. Thus, the optimal solution for ξ is derived as:

$$\xi_{ijl} = \phi(b_i^\top d_{\Pi_{il}}) - \alpha_{ij}. \tag{3.8}$$

Note that we only need to store all the predicted ratings $\mathbf{b}_i^\top\mathbf{d}_j$ and α_{ij} instead of ξ. Similiar to update ξ, by letting the derivation of problem (3.7) w.r.t. α_{ij} equals to zero, we can update α by

$$\alpha_{ij} = \frac{\frac{1}{2}(\frac{n_i-j+1}{2} - 1) + \sum_{l=j}^{n_i}\lambda(\xi_{ijl})\phi(b_i^\top d_{\Pi_{il}})}{\sum_{l=j}^{n_i}\lambda(\xi_{ijl})}. \tag{3.9}$$

We do not need to store $\lambda(\xi)$ either, since it can be computed on-the-fly in constant time when we cache these predicted ratings and α.

Learning B: We update $\mathbf{B}$ by fixing $\xi, \alpha, \mathbf{D}, \mathbf{X}, \mathbf{Y}$. Since the objective function in (3.7) is based on summing over independent users, we can update $\mathbf{B}$ by updating $\mathbf{b}_i$ in parallel according to

$$\begin{aligned}\underset{\mathbf{b}_i\in\{\pm 1\}^r}{\operatorname{argmin}} & \mathbf{b}_i^\top \sum_{j\in\Omega_i}\sum_{t=1}^{l}\frac{\mathbf{d}_j}{2} + \mathbf{b}_i^\top\Big(\sum_{j\in\Omega_i}\sum_{t=1}^{l}\lambda(\xi_{itl})\mathbf{d}_j\mathbf{d}_j^\top\Big)\frac{\mathbf{b}_i}{2r} \\ & - 2\mathbf{b}_i^\top\Big(\sum_{j\in\Omega_i}\sum_{t=1}^{l}\lambda(\xi_{itl})(\alpha_{it}-\frac{1}{2})\mathbf{d}_j\Big) \\ & - \mathbf{b}_i^\top(\sum_{j\in\Omega_i}\mathbf{d}_j) - 4r\beta_1\mathbf{b}_i^\top x_i,\end{aligned} \tag{3.10}$$

where Ω_i denotes the set of items rated by user i.

Due to the discrete constraints, the optimization is generally NP-hard, we adopt the bitwise learning method called Discrete Coordinate Descent (DCD) [Shen *et al.*, 2015, 2017] to update $\mathbf{b}_i$. In particular, denoting b_{ik} as the k-th bit of b_i and $b_{i\bar{k}}$ as the rest codes excluding b_{ik}. DCD updates b_{ik} while fixing $\mathbf{b}_{i\bar{k}}$. Thus, the updating rule for user binary code b_{ik} can be formulated as

$$b_{ik} \leftarrow sgn(O(\hat{b}_{ik}, b_{ik})), \tag{3.11}$$

where $\hat{b}_{ik} = \sum_{j\in\Omega_i}\sum_{t=1}^{l} 2\lambda(\xi_{itl})\Big(-\mathbf{d}_{j\bar{k}}^\top(\frac{\mathbf{b}_{i\bar{k}}}{2r})d_{jk} + (\alpha_{it} - \frac{1}{2})d_{jk}\Big) - \sum_{j\in\Omega_i}(\frac{l}{2}-1)d_{jk} + 4r\beta_1 x_{ik}$ and $O(x,y)$ is a function that $O(x,y) = x$ if $x \neq 0$ and $O(x,y) = y$ otherwise. We iteratively update each bit until the procedure converges with a set of better codes $\mathbf{b}_i$.

Learning D: Similarly, we can learn binary codes for item j by solving

$$\begin{aligned}\underset{\mathbf{d}_j\in\{\pm 1\}^r}{\operatorname{argmin}} & \mathbf{d}_j^\top \sum_{i\in\Omega_j}\sum_{t=1}^{l}\frac{\mathbf{b}_i}{2} + \mathbf{d}_j^\top\Big(\sum_{i\in\Omega_j}\sum_{t=1}^{l}\lambda(\xi_{itl})\mathbf{b}_i\mathbf{b}_i^\top\Big)\frac{\mathbf{d}_j}{2r} \\ & - 2\mathbf{d}_j^\top\Big(\sum_{i\in\Omega_j}\sum_{t=1}^{l}\lambda(\xi_{itl})(\alpha_{it}-\frac{1}{2})\mathbf{b}_i\Big) \\ & - \mathbf{d}_j^\top(\sum_{i\in\Omega_j}\mathbf{b}_i) - 4r\beta_2\mathbf{d}_j^\top y_j,\end{aligned} \tag{3.12}$$

where Ω_j denotes the set of users who rated the item j. Based on the DCD method, we update each bit of $\mathbf{d}_j$ according to

$$d_{jk} = sgn(O(\hat{d}_{jk}, d_{jk})), \tag{3.13}$$

where $\hat{d}_{jk} = \sum_{i\in\Omega_j}\sum_{t=1}^{l} 2\lambda(\xi_{itl})\Big(-\mathbf{b}_{i\bar{k}}^\top(\frac{\mathbf{d}_{j\bar{k}}}{2r})b_{ik} + (\alpha_{it} - \frac{1}{2})b_{ik}\Big) - \sum_{i\in\Omega_j}(\frac{l}{2}-1)b_{ik} + 4r\beta_2 y_{jk}$.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Learning $\mathbf{X}$ and $\mathbf{Y}$: When $\mathbf{B}$ fixed, learning $\mathbf{X}$ could be solved via optimizing the following problem:

$$\max_{\mathbf{X}} tr(\mathbf{B}^\top \mathbf{X}), \quad \mathbf{1}_m^\top \mathbf{X} = 0, (\mathbf{X})^\top \mathbf{X} = m\mathbf{I}_r.$$

It can be solved by the aid of SVD according to [Liu *et al.*, 2014a]. Let $\bar{\mathbf{B}}$ be a column-wise zero-mean matrix, where $\bar{B}_{ij} = B_{ij} - \frac{1}{m}\sum_i B_{ij}$. Assuming $\bar{\mathbf{B}} = \mathbf{P}_b \mathbf{\Sigma}_b (\mathbf{Q}_b)^\top$ as its SVD, where each column of $\mathbf{P}_b \in \mathbb{R}^{m\times r'}$ and $\mathbf{Q}_b \in \mathbb{R}^{r\times r'}$ represents the left and right singular vectors corresponding to r' non-zero singular values in the diagonal matrix $\mathbf{\Sigma}_b$. Since $\bar{\mathbf{B}}$ and $\mathbf{Q}_b$ have the same row, we have $\mathbf{1}^\top \mathbf{P}_b = 0$ due to $\mathbf{1}^\top \bar{\mathbf{B}} = 0$. Then we construct matrices $\hat{\mathbf{P}}_b$ of size $m\times(r-r')$ and $\hat{\mathbf{Q}}_b$ of size $r\times(r-r')$ by employing a Gram-Schmidt process such that $\hat{\mathbf{P}}_b^\top \hat{\mathbf{P}}_b = \mathbf{I}_{r-r'}$, $[\mathbf{P}_b\ \mathbf{1}]^\top \hat{\mathbf{P}}_b = 0$, and $\hat{\mathbf{Q}}_b^\top \hat{\mathbf{Q}}_b = \mathbf{I}_{r-r'}$, $[\mathbf{Q}_b\ \mathbf{1}]^\top \hat{\mathbf{Q}}_b = 0$. Now we obtain a closed-form update rule for $\mathbf{X}$:

$$\mathbf{X} \leftarrow \sqrt{m}[\mathbf{P}_b,\ \hat{\mathbf{P}}_b][\mathbf{Q}_b,\ \hat{\mathbf{Q}}_b]^\top. \tag{3.14}$$

In practice, to compute such an optimal $\mathbf{X}$, we perform the eigendecomposition over the small $r \times r$ matrix

$$\bar{\mathbf{B}}^\top \bar{\mathbf{B}} = [\mathbf{Q}_b\ \hat{\mathbf{Q}}_b] \begin{bmatrix} (\mathbf{\Sigma})^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} [\mathbf{Q}_b\ \hat{\mathbf{Q}}_b]^\top,$$

which provides $\mathbf{Q}_b, \hat{\mathbf{Q}}_b, \mathbf{\Sigma}$, and we can obtain $\mathbf{P}_b = \bar{\mathbf{B}}\mathbf{Q}_b\mathbf{\Sigma}^{-1}$. Then matrix $\hat{\mathbf{P}}_b$ can be obtained by the aforementioned Gram-Schmidt orthogonalization. Note that it requires $O(r^2 m)$ to perform SVD, Gram-Schimdt orthogonalization and matrix multiplication.

When $\mathbf{D}$ fixed, learning $\mathbf{Y}$ could be solved in a similar way:

$$\max_{\mathbf{Y}} tr(\mathbf{D}\mathbf{Y}^\top), \quad \mathbf{1}_n^\top \mathbf{Y} = 0, \mathbf{Y}^\top \mathbf{Y} = n\mathbf{I}_r.$$

Similarly, we can obtain an analytic solution:

$$\mathbf{Y} \leftarrow \sqrt{n}[\mathbf{P}_d,\ \hat{\mathbf{P}}_d][\mathbf{Q}_d,\ \hat{\mathbf{Q}}_d]^\top. \tag{3.15}$$

where each column of $\mathbf{P}_d$ and $\mathbf{Q}_d$ is the left and right singular vectors of $\bar{\mathbf{D}}$, respectively. $\hat{\mathbf{P}}_d$ are the left singular vectors corresponding to zero singular values of the $r \times r$ matrix $\bar{\mathbf{D}}^\top \bar{\mathbf{D}}$, and $\hat{\mathbf{Q}}_d$ are the vectors obtained via the Gram-Schimidt process.

4 Algorithmic Analysis

We discuss the initialization issues and time complexity in this section.

4.1 Initialization Note that the optimization problem in Eq. (3.7) involves a mixed-integer non-convex problem, a better initialization is important for faster convergence and better local optimal solution. In order to achieve an efficient initialization, we initialize $\mathbf{B}$, $\mathbf{D}$, $\mathbf{X}$ and $\mathbf{Y}$ by relaxing the binary constraints as

$$\begin{aligned} \min \sum_{\Pi\in S(R,\Omega)} -\sum_{i=1}^{m}\sum_{j=1}^{n_i} \log \frac{e^{\phi(\mathbf{u}_i^\top \mathbf{v}_{\Pi_{ij}})}}{\sum_{l=j}^{n_i} e^{\phi(\mathbf{u}_i^\top \mathbf{v}_{\Pi_{il}})}} \\ + \beta_3\|\mathbf{U}\|_F^2 + \beta_4\|\mathbf{V}\|_F^2 - 2\beta_1 tr(\mathbf{U}^\top \mathbf{X}) - 2\beta_2 tr(\mathbf{V}^\top \mathbf{Y}) \\ \text{s.t.} \mathbf{1}_m^\top \mathbf{X} = 0, \mathbf{1}_n^\top \mathbf{Y} = 0, \mathbf{X}^\top \mathbf{X} = m\mathbf{I}_r, \mathbf{Y}^\top \mathbf{Y} = n\mathbf{I}_r. \end{aligned} \tag{4.16}$$

where $\phi(x) = 1/(1 + exp(-x))$ is the sigmoid function.

We first initialize the real-valued matrix $\mathbf{U}$ and $\mathbf{V}$ randomly and find the feasible solution for $\mathbf{X}$ and $\mathbf{Y}$ according to our proposed learning method w.r.t. $\mathbf{X}$ and $\mathbf{Y}$. Then the alternating optimization is conducted by updating $\mathbf{U}$ and $\mathbf{V}$ with traditional gradient descent method and updating $\mathbf{X}$ and $\mathbf{Y}$ according to our proposed learning method. Once we obtain the solution $(\mathbf{U}^0, \mathbf{V}^0, \mathbf{X}^0, \mathbf{Y}^0)$, we can initialize our algorithm as:

$$\mathbf{B} \leftarrow \text{sgn}(\mathbf{U}^0), \mathbf{D} \leftarrow \text{sgn}(\mathbf{V}^0), \mathbf{X} \leftarrow \mathbf{X}^0, \mathbf{Y} \leftarrow \mathbf{Y}^0. \tag{4.17}$$

The effectiveness of the proposed initialization is illustrated in Figure 1. We can see that without initialization, DLCF algorithm needs more than 10 iterations to converge while with initialization, it only takes $5 \sim 7$ iterations and achieves better ranking performance.

Algorithm 1 DLCF Algorithm

Input: Observed user-item ratings $\mathbf{R} \in \mathbb{R}^{m\times n}$
Output: $\mathbf{B} \in \{\pm 1\}^{r\times m}, \mathbf{D} \in \{\pm 1\}^{r\times n}$
Parameters: number of components G, code length r, regularization coefficient α_1, α_2, bandwidth parameter h
Initialize $\mathbf{B}, \mathbf{D}$ and $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m\times n}$ according to (4.17).
while not converged **do**
  Update α and ξ according to (3.9) and (3.8), repectively.
  for $i = 1, \cdots, m$ **do**
    Update $\mathbf{b}_i$ bit-by-bit according to (3.11)
  end for
  for $j = 1, \cdots, n$ **do**
    Update $\mathbf{d}_j$ bit-by-bit according to (3.13).
  end for
  Update $\mathbf{X}$ and $\mathbf{Y}$ according to (3.14) and (3.15).
end while
Return $\mathbf{B}, \mathbf{D}$.

4.2 Computation Complexity for Model Training For each iteration, the computation cost for updating $\mathbf{B}$ and $\mathbf{D}$ is $O(\#iter m\bar{n}r^2)$, where $\bar{n}$ is the average number of observed items for each user and $\#iter$ is usually $5 \sim 7$ in practice. The computation cost for updating α and ξ is $O(m\bar{n})$. For updating $\mathbf{X}$ and $\mathbf{Y}$, it takes $O(r^2 m)$ and $O(r^2 n)$, respectively. Suppose the entire algorithm requires T iterations for convergence, the overall time complexity

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

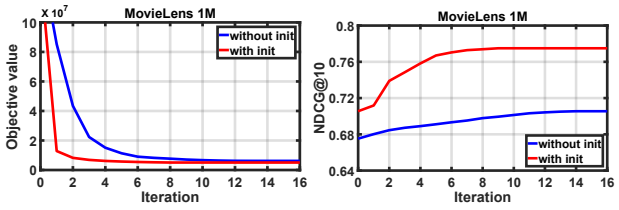

Figure 1: Convergence curve of the objective values and NDCG@10 of DLCF with/without initialization on the Movielens 1M dataset.

for training DLCF is $O(T(m\bar{n}r^2 + mr^2 + nr^2 + m\bar{n}))$. In summary, training DLCF is efficient even considering a listwise ranking modal since it scales linearly with the size of the data. Compared to the time complexity of the discrete pointwise CF method in [Zhang *et al.*, 2016], which is $O(T(m\bar{n}r^2+mr^2+nr^2))$, DLCF only requires additional cost for updating the variational parameter ξ and α.

5 Experiments

As the key contribution of this work is the design of DLCF for fast and accurate recommendation, we conduct experiments to answer the following research questions:

RQ1: Compared with state-of-the-art binarized collaborative filtering methods, how does DLCF perform in terms of accuracy? How does the learning method benefit from joint discrete optimization?

RQ2: How efficient is DLCF as compared to the real-valued version of SQL-Rank?

RQ3: What's the gap between the original likelihood loss in (3.2) and its quadratic upper bound approximation used in (3.4)? How about the convergence of Algorithm 1?

Dataset	#Ratings	#Users	#Items	#density
Movielens 1M	1,000,209	6040	3900	4.2%
Yelp	696,865	25,677	25,815	0.11%
Netflix	100,480,507	480,189	17,770	1.18%

Table 1: Summary of datasets in our experiments.

Datasets and Settings. We experimented with three publicly accessible datasets: Movielens 1M [1], Yelp [2] and a subset of Netflix. All of these ratings range from 0 to 5. We followed the conventional filtering strategy by removing users and items having less than 10 ratings [Rendle *et al.*, 2009]. Table 1 summaries the experimental datasets. For each user, we randomly sample 70% ratings as training data and the rest 30% as test data. We repeated a random split 5 times for each dataset and computed the average results of each algorithm over 5 runs. All the experiments were conducted on a computer equipped with an Intel(R) Core(TM) i5-7200U CPU @2.50GHZ, 16GB RAM and 64-bit Windows 10 operating system.

Parameter Settings and Performance Metrics. The code length in DLCF varies in range $\{8, 16, 24, 32\}$. The two hyper-parameters β_1 and β_2 are tuned within $\{10^{-4}, ..., 10^{-1}\}$. Grid search is used to choose the best parameters on the training split. We evaluate our proposed algorithms by NDCG@10. A higher NDCG value reflects a better accuracy.

Baseline Methods. We compare DLCF with the state-of-the-art BCF methods and real-valued LCF methods:

- **SQL-Rank** is the state-of-the-art real-valued LCF method [Wu *et al.*, 2018].
- **DRMF** is the state-of-the-art BCF method [Zhang *et al.*, 2018] with pairwise ranking loss.
- **DCF** is the first BCF method [Zhang *et al.*, 2016] that directly tackles a discrete optimization problem for pointwise loss.
- **PPH** is a two-stage hashing based method [Zhang *et al.*, 2014] with a relaxation stage and a quantization stage.
- **BSQL-Rank** is the SQL-Rank results with round-off binary quantization.
- **DLCFinit** is the initialization problem of DLCF in Eq. (4.16). We used BSQL-Rank and DLCFinit as baselines to validate the effectiveness of the proposed joint optimization.

5.1 Accuracy (RQ1) For fair comparison, the code length and latent vector dimension of all the methods are set to be identical, so that the performance gain is not caused by increasing model complexity. In Figure 2 and Figure 3, we compare DLCF with the baseline methods whose code length varies from 8 to 32 in terms of NDCG on three datasets. We can draw the following observations:

- We find that DLCF considerably outperforms PPH, DCF, DRMF which are the state-of-the-art binarized CF methods. This observation verifies the remarkable advantage of utilizing listwise ranking loss over pointwise and pairwise loss in discrete optimization of CF model. Among baseline methods, DCF, DRMF consistently outperforms PPH. This is consistent with the findings in [Zhang *et al.*, 2016, 2018] that the performance of direct discrete optimization could surpass that of the two-stage methods.
- Besdies, DLCF shows very competitive performance compared with the real-valued listwise CF method, SQL-Rank. As the bit size increases, the performance

[1] http://grouplens.org/datasets/movielens
[2] http://www.yelp.com/dataset

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

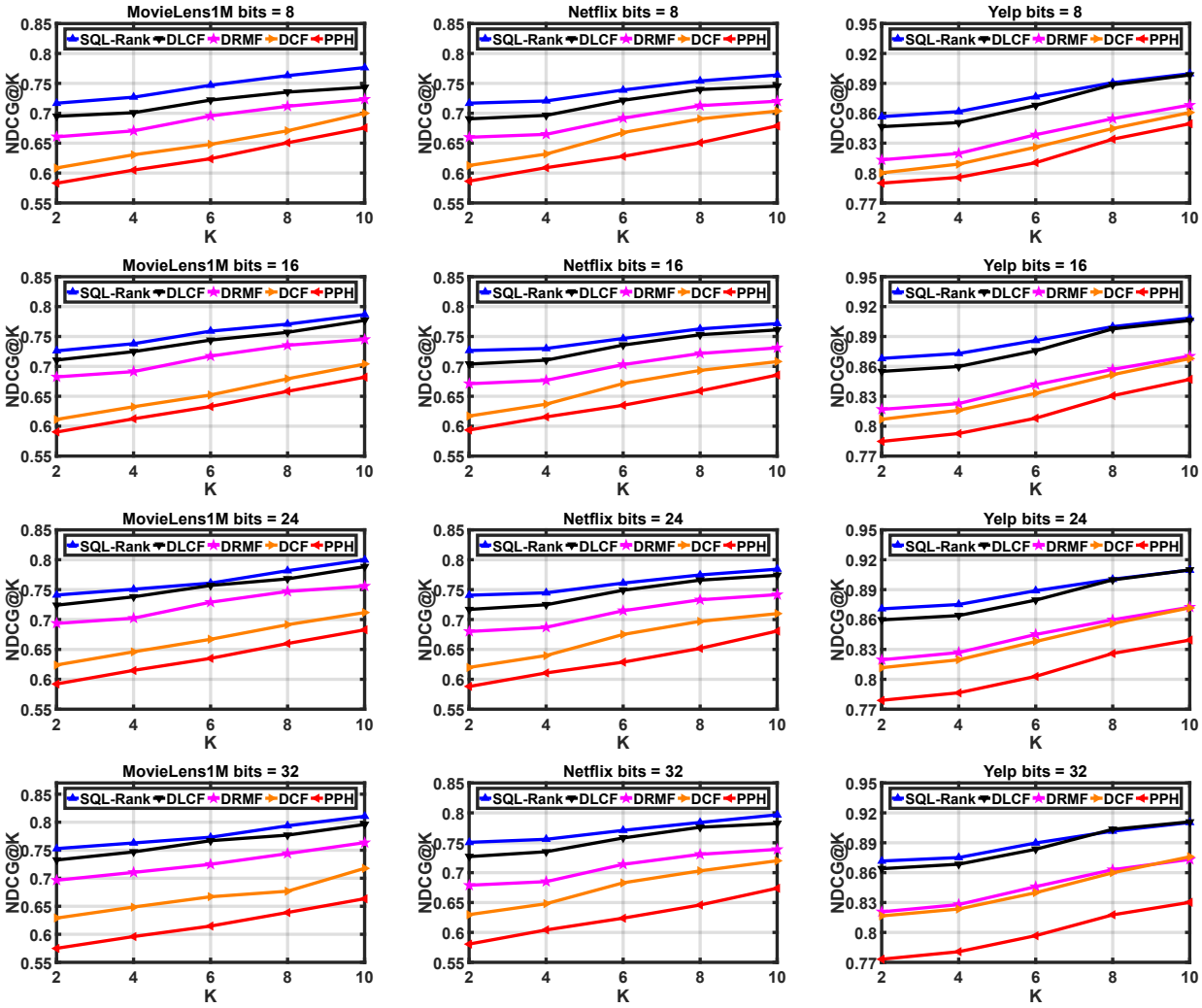

Figure 2: Recommendation performance comparison of NDCG@10 w.r.t. different code length.

gap between DLCF and SQL-Rank shrinks quickly. It could even achieve the same performance with SQL-Rank on Yelp dataset which is very sparse. One possible reason is that the higher sparsity of the dataset makes real-valued vectors in SQL-Rank easy to overfit, whereras the binarized codes in DLCF could alleviate this issue since it has a much lower model complexity.

- Lastly, DLCF outperforms DLCFinit and BSQL-Rank, which again verifies the superiority of the proposed joint discrete optimization over the two-stage optimization method in listwise CF model.

5.2 Efficiency (RQ2) Table 2 shows the retrieval time of DLCF (binary codes) and SQL-Rank (real-valued vectors) to generate the top-k item list of all users by linearly scanning the whole corpus. It is obvious that DLCF outperforms SQL-Rank by an order of magnitude, which indicates the great benefit of binarizing the real-valued parameters in listwise CF model. This makes DLCF a more suitable model for large-scale recommender systems where the retrieval time for items is restricted within a limited time quota. Table 3 demonstrates the memory usage of DLCF and SQL-Rank. We can find that DLCF considerably reduces the memory storage of the model for 37 times within the same dimension. These results imply that DLCF can adapt to some resource-limited scenarios.

5.3 Effectiveness of the Quadratic Upper Bound Approximation (RQ3) Notice that the objective of DLCF in (3.4) minimizes a variational quadratic upper bound instead of directly optimizing the listwise loss function in (3.2). In this section, we test the gap between the DLCF loss (based on the upper bound) and the original listwise loss in the training procedure. In Figure 4, we can find that the gap between

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

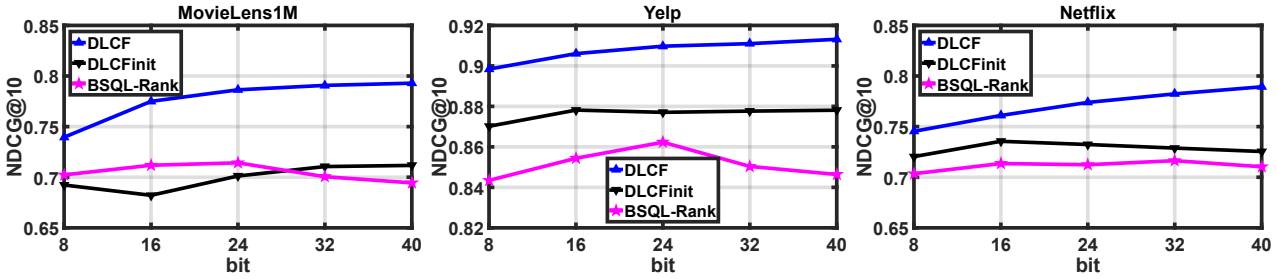

Figure 3: Recommendation performance comparison between joint optimization and two-stage method.

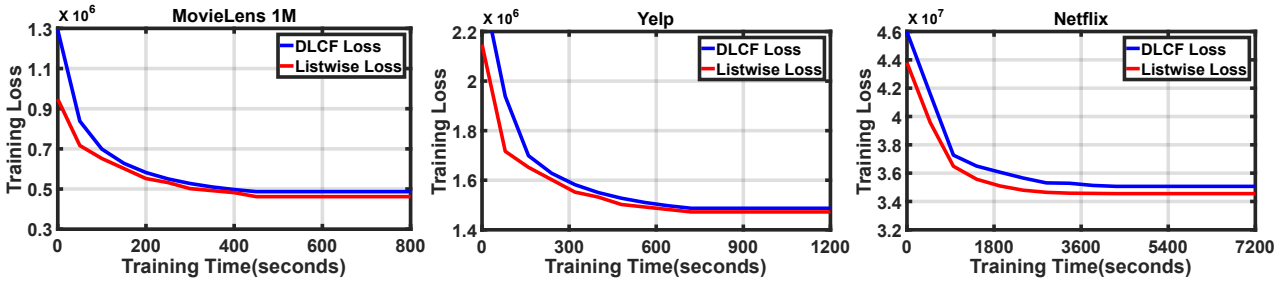

Figure 4: The training loss comparison of DLCF and listwise loss.

Dataset	SQL-Rank		DLCF
	Time	Speedup	Time
Movielens 1M	1.66 min	×4.74	0.35 min
Netflix	266.39 min	×4.97	53.61 min
Yelp	41.95 min	×5.11	8.20 min

Table 2: Retrieval time of DLCF and SQL-Rank with the same code length (40) on three datasets.

Dataset	SQL-Rank		DLCF
	Memory	Reduction	Memory
Movielens 1M	2.92 MB	×37	80.6 KB
Netflix	40.1 MB	×37	1.07 MB
Yelp	15.1 MB	×37	416 KB

Table 3: Memory usage of DLCF and SQL-Rank with the same code length (40) on three datasets.

the two loss functions is relatively small, which indicates the upper bound approximation is a good approximation of the original listwise loss in our model. Therefore, it can guarantee a good recommendation accuracy. Besides, we can conclude that the proposed optimization method could converge, which further comfirm the correctness of the proposed optimization method.

6 Conclusion

In this paper, we propose a discrete listwise collaborative filtering method to learn informative yet compact binary codes for users and items, which is aligned with the ultimate goals of recommender systems. DLCF directly addresses the discrete optimization problem by iteratively solving several mixed-integer programming subproblems. Extensive experiments on three real-world datasets demonstrate the advantages of our proposed method against several competitive baselines in terms of recommendation accuracy, retrieval cost and storage cost.

Acknowledgements

Xin Wang is supported by the National Key Research and Development Program of China (No. 2020AAA0107800, 2020AAA0106300, 2018AAA0102000).

References

Guillaume Bouchard. Efficient bounds for the softmax function and applications to approximate inference in hybrid models. In *NIPS 2007 workshop for approximate Bayesian inference in continuous/hybrid systems*, 2007.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization:scalable online collaborative filtering. In *International Conference on World Wide Web*, pages 271–280, 2007.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 09/13/21 to 218.247.253.234 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Wang-Cheng Kang and Julian McAuley. Candidate generation with binary codes for large-scale top-n recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1523–1532, 2019.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

Defu Lian, Rui Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. Discrete content-aware matrix factorization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 325–334. ACM, 2017.

Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, pages 3419–3427, 2014.

Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. Collaborative hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154, 2014.

Han Liu, Xiangnan He, Fuli Feng, Liqiang Nie, Rui Liu, and Hanwang Zhang. Discrete factorization machines for fast feature-based recommendation. *arXiv preprint arXiv:1805.02232*, 2018.

Chenghao Liu, Tao Lu, Xin Wang, Zhiyong Cheng, Jianling Sun, and Steven CH Hoi. Compositional coding for collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 145–154, 2019.

Chenghao Liu, Xin Wang, Tao Lu, Wenwu Zhu, Jianling Sun, and Steven Hoi. Discrete social recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 208–215, 2019.

John I Marden. *Analyzing and modeling rank data*. Chapman and Hall/CRC, 2014.

Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.

Fumin Shen, Yadong Mu, Yang Yang, Wei Liu, Li Liu, Jingkuan Song, and Heng Tao Shen. Classification by retrieval: Binarizing data and classifier. 2017.

Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):2393–2406, 2012.

Shuaiqiang Wang, Shanshan Huang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. Ranking-oriented collaborative filtering: A listwise approach. *ACM Transactions on Information Systems (TOIS)*, 35(2):10, 2016.

Liwei Wu, Cho-Jui Hsieh, and James Sharpnack. Sql-rank: A listwise approach to collaborative ranking. *arXiv preprint arXiv:1803.00114*, 2018.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. Preference preserving hashing for efficient recommendation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 183–192. ACM, 2014.

Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 325–334. ACM, 2016.

Yan Zhang, Defu Lian, and Guowu Yang. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *AAAI*, pages 1669–1675, 2017.

Yan Zhang, Haoyu Wang, Defu Lian, Ivor W Tsang, Hongzhi Yin, and Guowu Yang. Discrete ranking-based matrix factorization with self-paced learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2758–2767. ACM, 2018.

Ke Zhou and Hongyuan Zha. Learning binary codes for collaborative filtering. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 498–506. ACM, 2012.

Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. Joint optimization of tree-based index and deep model for recommender systems. In *Advances in Neural Information Processing Systems*, pages 3973–3982, 2019.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited