Discrete Listwise Personalized Ranking for Fast Top-N Recommendation with Implicit Feedback

Fangyuan Luo , Jun Wu* , Tao Wang
School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
{fangyuanluo, wuj, twang}@bjtu.edu.cn

Abstract

We address the efficiency problem of personalized ranking from implicit feedback by hashing users and items with binary codes, so that top-N recommendation can be fast executed in a Hamming space by bit operations. However, current hashing methods for top-N recommendation fail to align their learning objectives (such as pointwise or pairwise loss) with the benchmark metrics for ranking quality (e.g. Average Precision, AP), resulting in sub-optimal accuracy. To this end, we propose a Discrete Listwise Personalized Ranking (DLPR) model that optimizes AP under discrete constraints for fast and accurate top-N recommendation. To resolve the challenging DLPR problem, we devise an efficient algorithm that can directly learn binary codes in a relaxed continuous solution space. Specifically, theoretical analysis shows that the optimal solution to the relaxed continuous optimization problem is exactly the same as that of the original discrete DLPR problem. Through extensive experiments on two real-world datasets, we show that DLPR consistently surpasses state-of-the-art hashing methods for top-N recommendation.

1 Introduction

Top-N recommendation is a fundamental task across many Internet services, which aims at discovering a short list of items that are likely to be interacted by users. However, it is challenging to rapidly identify top-N items from a large corpus due to the stringent response requirement of Web services. Consequently, a "recall-and-ranking" recommendation architecture is widely adopted in industry [Covington *et al.*, 2016]. That is, a scalable recall model first identifies a relatively small set of candidate items, and then a fully-blown ranking model refines the rank list of candidate items. This work mainly concentrates on the recall model that scales well to a large volume of items.

In the recall stage, embeddings of users and items are often pre-computed via matrix factorization (MF) [Koren *et al.*, 2009] or deep neural networks (DNNs) [Zhang *et al.*, 2019] ; after that, candidates generation naturally falls into a similarity search problem, i.e. finding top-N similar items queried by a user based on pre-computed embeddings. However, due to the ever-growing volume of items, similarity search has been a major efficiency bottleneck for top-N recommendation. Fortunately, hashing has been shown as a promising solution to fast similarity search [Wang *et al.*, 2012], the core idea of which is encoding real-valued embeddings into binary codes, so that user-item similarities can be efficiently calculated in a Hamming space by bit operations. During the past decade, a surge of efforts have been made in hashing-based recommendation (HR), most of which learn binary codes from *explicit* feedback (e.g., ratings) [Zhou and Zha, 2012; Zhang *et al.*, 2014; Zhang *et al.*, 2016; Zhang *et al.*, 2018b; Liu *et al.*, 2019; Wu *et al.*, 2020; Luo *et al.*, 2021; Liu *et al.*, 2021; Hansen *et al.*, 2021]. However, their applicability is limited because explicit feedback is fairly expensive to obtain. Hence a recent trend is to perform hash for recommendation with *implicit* feedback (such as purchasing and clicking) [Zhang *et al.*, 2017; Zhang *et al.*, 2018a; Zhang *et al.*, 2020; Lian *et al.*, 2021], which formulate their learning tasks as Binary Quadratic Programming (BQP) problems and solve them by Discrete Coordinate Descent (DCD) which is the *de facto* choice for solving BQP problems [Shen *et al.*, 2015]. We refer to the HR schemes using *explicit* and *implicit* feedback as the HRex and the HRim respectively, and the latter is our focus in this work.

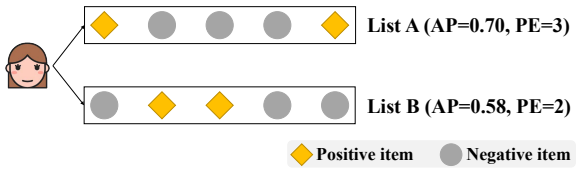


Figure 1: An illustration of the inconsistency between pairwise error (PE) and AP metric (cf. Eq.(2)), where PE is defined as the count of item pairs ranked incorrectly (i.e. a negative item is ranked in front of a positive one) in a rank list.

However, existing HRim schemes mainly focus on optimizing a pairwise (even a pointwise[1]) objective, which is sub-

*Corresponding author.

[1]Given implicit feedback, pointwise objective often performs worse than pariwise one in terms of top-N recommendation tasks [Rendle *et al.*, 2009].

optimal for personalized ranking. It is well-known that the benchmark metric for ranking quality is Average Precision (AP) (or its generalized variant, Normalized Discounted Cumulative Gain (NDCG), which includes non-binary relevance judgements)[2][Brown *et al.*, 2020]. However, the core idea of pairwise objective is inconsistent with AP. Illustrated by Figure 1, given two rank lists A and B, A achieves better AP than B (0.7 vs 0.58), but B is with smaller pairwise error than A (2 vs 3). An explanation is that AP imposes larger penalty on the top-ranked negative items, while pairwise loss equally treats all negative items in the rank list.

In this paper, we propose a novel HRim approach, dubbed Discrete Listwise Personalized Ranking (DLPR), which optimizes AP under discrete constraints for better recall accuracy. Due to the sorting function used by AP, DLPR cannot be formulated as a BQP problem. Consequently, its optimization, e.g. with DCD, is notoriously difficult. To this end, we develop an alternative optimization algorithm that can directly learn binary codes in a relaxed continuous solution space. Theoretical analysis shows that the optimal solution to the relaxed continuous optimization problem is the same as that of the original discrete DLPR problem. Our code and supplementary file are available at https://github.com/LFY123456/DLPR.

Specifically, our contributions are briefly summarized as:

- We propose a novel DLPR model for better preserving the listwise user-item relationship while guaranteeing recommendation efficiency. To the best of our knowledge, this is the first work to perform hashing technique on optimizing AP for top-N recommendation.
- We devise an efficient algorithm for solving DLPR, and its convergence is rigorously guaranteed.
- Extensive experiments on two real-world datasets demonstrate that DLPR consistently outperforms state-of-the-art HRim methods.

2 Related Work

In this section, we mainly review related work on HR. For comprehensive reviews of hashing techniques, please refer to [Wang *et al.*, 2016; Wang *et al.*, 2018a].

The first hashing solution to recommendation is proposed by [Zhou and Zha, 2012], named Binary Code for Collaborative Filtering (BCCF), which conducts Iterative Quantization (ITQ) to generate binary codes from rotated real latent representations. To derive compact binary codes, the balanced constraints are imposed on the real latent representations. However, according to [Zhang *et al.*, 2014], hashing only preserves similarity between users and items rather than inner product based preference, leading to the decrease of recommendation accuracy. Thus, they proposed a Preference Preserving Hashing (PPH) method to preserve users' preference over items. These work can be summarized as two-stage method which first learn real-valued latent features for users and items and then perform quantization on them to get hash codes. However, two-stage methods suffer from a large quantization loss according to [Zhang *et al.*, 2016].

Direct binary codes learning is becoming popular in order to decrease quantization loss, called One-stage Learning (OL). In detail, OL strategy first formulates learning objectives as BQP problems and then solved by DCD. To learn informative and compact codes, balanced and de-correlated constraints are imposed on discrete optimization process [Zhang *et al.*, 2016]. A few fellow-up studies along this line learn binary codes for fast recommendation [Liu *et al.*, 2019; Wu *et al.*, 2020; Luo *et al.*, 2021; Liu *et al.*, 2021; Hansen *et al.*, 2021]. Note that these methods are deployed in explicit scenarios, i.e., HRex methods, where explicit feedbacks are expensive to obtain. Hence, some studies turn to optimize pointwise or pairwise loss in implicit scenarios aiming to achieve better top-N recommendation accuracy, which give birth to HRim methods [Zhang *et al.*, 2017; Zhang *et al.*, 2018a; Zhang *et al.*, 2020; Lian *et al.*, 2021]. However, the learning objective of HRim shows the inconsistency with evaluation criterion AP which is the benchmark metric for ranking quality. Thus, current HRim methods are still sub-optimal for top-N recommendation.

3 Preliminaries

Throughout this paper, we use bold uppercase and lowercase letters as matrices and vectors, respectively. All of vectors in this paper denote column vectors. Non-bold letters represent scalars. We denote $tr(\bullet)$ as the trace of a matrix. In this work, we focus on optimizing AP under discrete constraints based on matrix factorization (MF) for top-N recommendation.

Let $\mathbf{Y} \in \{0,1\}^{m\times n}$ denote the implicit feedback matrix, where the element y_{ui} indicates whether user u has interacted with item i (1) or not (0). MF associates each user and item with a real-valued vector of latent features. Supposing $\mathbf{p}_u \in \mathbb{R}^f$ and $\mathbf{q}_i \in \mathbb{R}^f$ are the uth user vector and the ith item vector, respectively, MF estimates the interaction $\hat{y}_{ui}$ as the inner product of $\mathbf{p}_u$ and $\mathbf{q}_i$:

$$\hat{y}_{ui} = \mathbf{p}_u^T \mathbf{q}_i. \tag{1}$$

Based on user u's preference over the items, then we can generate a recommendation list $\pi_{\hat{\mathbf{y}}_u} = sort(\hat{\mathbf{y}}_u)$ by sorting the items in a descending order according to their predicted scores $\hat{\mathbf{y}}_u$. Then, the AP score of the list is defined as:

$$AP_u@K = \frac{\sum_{k=1}^{K} \frac{y_{u\pi_{\hat{\mathbf{y}}_u}(k)}}{k} \sum_{i=1}^{k} y_{u\pi_{\hat{\mathbf{y}}_u}(i)}}{\sum_{k=1}^{K} y_{u\pi_{\hat{\mathbf{y}}_u}(k)}}, \tag{2}$$

where $\pi_{\hat{\mathbf{y}}_u}(k)$ denotes the item at rank k.

4 Discrete Listwise Personalized Ranking

We first present the learning objective of DLPR and then describe its detailed optimization process, which is the key technical difficulty of this paper. Finally, we give the convergence analysis of our proposed DLPR algorithm.

4.1 Model Formulation

In this paper, we are interested in hashing users and items into binary codes for fast recommendation, where user-item

[2]Given binary judgements, NDCG is almost the same with AP, except that they utilize different discount functions.

similarity calculation can be efficiently conducted in Hamming space. Denote $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_m] \in \{\pm 1\}^{f \times m}$ and $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_n] \in \{\pm 1\}^{f \times n}$ respectively as f-length user and item hash codes, the Hamming similarity between $\mathbf{b}_u$ and $\mathbf{d}_i$ is defined as:

$$\hat{y}_{ui} = \frac{1}{f}\sum_{r=1}^{f} \mathbb{I}(b_{ur} = d_{ir}) = \frac{1}{2} + \frac{1}{2f}\mathbf{b}_u^T \mathbf{d}_i. \tag{3}$$

Substituting Eq.(3) into Eq.(2), the objective of our proposed method can be formulated as:

$$\begin{aligned}\mathcal{L}(\mathbf{B}, \mathbf{D}) &= 1 - \frac{1}{m}\sum_{u=1}^{m} AP_u@K \\ &= 1 - \frac{1}{m}\sum_{u=1}^{m} \frac{\sum_{k=1}^{K} \frac{y_{u\pi_{\hat{\mathbf{y}}_u}(k)}}{k} \sum_{i=1}^{k} y_{u\pi_{\hat{\mathbf{y}}_u}(i)}}{\sum_{k=1}^{K} y_{u\pi_{\hat{\mathbf{y}}_u}(k)}}, \\ s.t. &\mathbf{B} \in \mathcal{D}_{\mathbf{B}}, \mathcal{D}_{\mathbf{B}} := \{\mathbf{B} : \mathbf{B} \in \{\pm 1\}^{f \times m}\}, \\ &\mathbf{D} \in \mathcal{D}_{\mathbf{D}}, \mathcal{D}_{\mathbf{D}} := \{\mathbf{D} : \mathbf{D} \in \{\pm 1\}^{f \times n}\}. \end{aligned} \tag{4}$$

Due to discrete constraints, optimizing Eq.(4) is a highly challenging task as illustrated in Section 1. Thus, we introduce a new learning objective that allows DLPR to be solved in a computationally tractable way. The basic idea is to transform Eq.(4) into a solvable concave optimization problem with the relaxed continuous solution space. As we know, the optimal solution of a concave function is necessarily located at a boundary of the convex set [Rockafellar, 1997].

To achieve this, we first relax the discrete solution space $\mathcal{D}_{\mathbf{B}} := \{\mathbf{B} : \mathbf{B} \in \{-1, 1\}^{f \times m}\}$ and $\mathcal{D}_{\mathbf{D}} := \{\mathbf{D} : \mathbf{D} \in \{-1, 1\}^{f \times n}\}$ to its convex hull $\mathcal{P}_{\mathbf{B}} := \{\mathbf{B} : \mathbf{B} \in [-1, 1]^{f \times m}\}$ and $\mathcal{P}_{\mathbf{D}} := \{\mathbf{D} : \mathbf{D} \in [-1, 1]^{f \times n}\}$ respectively. Thus the objective function is continuous while non-differentiable since the computation of AP relies on the sorting operator. Therefore, we replace the sorting function by a softmax function inspired by [Grover *et al.*, 2019] to smooth the sorting operator which enable direct optimization of AP.

Proposition 1. *Denote $\mathbf{s} = [s_1, s_2, \cdots, s_l]^T$ be a real-valued score vector of length l and $\mathbf{A}_{\mathbf{s}}$ be the matrix of absolute pairwise differences of the elements of $\mathbf{s}$ such that $A_{\mathbf{s}}[i, j] = |s_i - s_j|$. Let $\pi_{\mathbf{s}}$ be the permutation of sorting $\mathbf{s}$ in a descending order. Then the permutation matrix $\mathbf{Z}^{(\pi_{\mathbf{s}})}$ can be represented as follows:*

$$Z_{i,j}^{(\pi_{\mathbf{s}})} = \begin{cases} 1, & j = argmax[(l + 1 - 2i)\mathbf{s} - \mathbf{A}_{\mathbf{s}}\mathbf{1}]; \\ 0, & otherwise. \end{cases} \tag{5}$$

Proof. See A.1 in supplementary file. □

Since the argmax operation is non-differentiable which prohibits the direct use of Eq.(5) for gradient computation. Instead, we replace the argmax operator with softmax to obtain a continuous relaxation $\widetilde{\mathbf{Z}}^{(\pi_{\mathbf{s}}, \tau)}$. In particular, the ith row of $\widetilde{\mathbf{Z}}^{(\pi_{\mathbf{s}}, \tau)}$ is given by:

$$\widetilde{\mathbf{Z}}_i^{(\pi_{\mathbf{s}}, \tau)} = softmax[\tau^{-1}((l + 1 - 2i)\mathbf{s} - \mathbf{A}_{\mathbf{s}}\mathbf{1})], \tag{6}$$

where $\tau > 0$ is a temperature parameter, which controls the approximation degree between softmax operation and argmax operation. Substituting $\widetilde{\mathbf{Z}}_i^{(\pi_{\mathbf{s}}, \tau)}$ into Eq.(4), the objective function of DLPR can be formulated as:

$$\begin{aligned}\mathcal{L}(\mathbf{B}, \mathbf{D}) &= 1 - \frac{1}{m}\sum_{u=1}^{m} \widetilde{AP}_u@K(\mathbf{y}_u, \pi_{\hat{\mathbf{y}}_u, \tau}) \\ &= 1 - \frac{1}{m}\sum_{u=1}^{m} \frac{\sum_{k=1}^{K} \frac{\widetilde{\mathbf{Z}}_k^{\pi_{\hat{\mathbf{y}}_u, \tau}} \mathbf{y}_u}{k} \sum_{i=1}^{k} \widetilde{\mathbf{Z}}_i^{\pi_{\hat{\mathbf{y}}_u, \tau}} \mathbf{y}_u}{\sum_{k=1}^{K} \widetilde{\mathbf{Z}}_k^{\pi_{\hat{\mathbf{y}}_u, \tau}} \mathbf{y}_u}, \\ s.t. &\mathbf{B} \in \mathcal{P}_{\mathbf{B}}, \mathcal{P}_{\mathbf{B}} := \{\mathbf{B} : \mathbf{B} \in [-1, 1]^{f \times m}\}, \\ &\mathbf{D} \in \mathcal{P}_{\mathbf{D}}, \mathcal{P}_{\mathbf{D}} := \{\mathbf{D} : \mathbf{D} \in [-1, 1]^{f \times n}\}. \end{aligned} \tag{7}$$

Specifically, there is a subtle connection between the true AP and its smooth version $\widetilde{AP}$, described by Proposition 2.

Proposition 2. *The $\widetilde{AP}$ is approximately equal to the AP, and the proximity between them depends on the value of τ. If $\tau \to 0^+$, the $\widetilde{AP}$ is a tight approximation to AP, vice versa.*

Proof. See A.2 in supplementary file. □

It is worth noting that our model is deployed in the *continuous space* and directly learn optimal *discrete solution* based on thought that the optimal solution of a concave function is necessarily located at a boundary of the convex set. More importantly, theoretical proofs show that the optimal discrete solution to Eq.(7) is exactly that of original discrete problem.

4.2 Optimization

We employ alternating optimization strategy to solve two subproblems for DLPR model in Eq.(7), taking turns to update each of $\mathbf{B}$ and $\mathbf{D}$, given the other fixed. Next, we elaborate on how to solve each of the subproblems.

B-subproblem

In this subproblem, we aim to optimize $\mathbf{B}$ with fixed $\mathbf{D}$. Specifically, we propose to use Graduated NonConvexity and Concavity Procedure (GNCCP) [Liu and Qiao, 2014; Wang *et al.*, 2018b] which obtains optimal discrete solution from the continuous vector space, to approximately solve the subproblem as follows:

$$\begin{aligned}\mathcal{L}_\zeta(\mathbf{B}) = &\begin{cases} (1 - \zeta)\mathcal{L}(\mathbf{B}) + \zeta tr(\mathbf{B}^T\mathbf{B}), 1 \geq \zeta \geq 0; \\ (1 + \zeta)\mathcal{L}(\mathbf{B}) + \zeta tr(\mathbf{B}^T\mathbf{B}), 0 \geq \zeta \geq -1, \end{cases} \\ &s.t., \mathbf{B} \in \mathcal{P}_{\mathbf{B}}, \mathcal{P}_{\mathbf{B}} := \{\mathbf{B} : \mathbf{B} \in [-1, 1]^{f \times m}\}. \end{aligned} \tag{8}$$

Actually, Eq.(8) realizes the convex and concave relaxation of Eq.(7) by knobbing the parameter ζ. With the decreasing of ζ from 1 to -1, implying that the objective function $\mathcal{L}_\zeta(\mathbf{B})$ becomes gradually from $tr(\mathbf{B}^T\mathbf{B})$ to $\mathcal{L}(\mathbf{B})$ (graduated non-convexity) and finally to $-tr(\mathbf{B}^T\mathbf{B})$ (graduated concavity). For each currently fixed ζ, $\mathcal{L}_\zeta(\mathbf{B})$ can be minimized by the Frank-Wolfe algorithm [Marguerite Frank, 1956], using the minimum of the previous $\mathcal{L}_\zeta(\mathbf{B})$ as the starting point. In detail, we update $\mathbf{B}$ according to:

$$\arg\min_{\mathbf{B}^*} tr(\nabla\mathcal{L}_\zeta(\mathbf{B})^T\mathbf{B}^*), \tag{9}$$

where the gradient $\nabla\mathcal{L}_\zeta(\mathbf{B})$ takes the form:

$$\nabla\mathcal{L}_\zeta(\mathbf{B}) = \begin{cases}(1-\zeta)\nabla\mathcal{L}(\mathbf{B}) + 2\zeta\mathbf{B}, & 1 \geq \zeta \geq 0;\\ (1+\zeta)\nabla\mathcal{L}(\mathbf{B}) + 2\zeta\mathbf{B}, & 0 \geq \zeta \geq -1.\end{cases} \quad (10)$$

And the linear programming problem in Eq.(9) can be solved based on the following rule: $\mathbf{B}^* = -sgn(\nabla\mathcal{L}_\zeta(\mathbf{B}))$. Next, we conduct linear search to find an update direction to optimize $\mathbf{B}$ until it converges. Then iteratively, given the optimization step d, we find a local minimum of $\mathcal{L}_{\zeta-d}(\mathbf{B})$ given a local minimum of $\mathcal{L}_\zeta(\mathbf{B})$ by performing a local optimization of $\mathcal{L}_{\zeta-d}(\mathbf{B})$ starting from the local minimum of $\mathcal{L}_\zeta(\mathbf{B})$. Repeating this iterative process for d small enough, we obtain a path of solutions $\{\mathbf{B}^*(\zeta)\}$ where $\mathbf{B}^*(\zeta)$ is a local minimum of $\mathcal{L}_\zeta(\mathbf{B})$ for all $\zeta \in [-1, 1]$. We finally output $\mathbf{B}^*(\zeta^*) \in \mathcal{D}$ as an approximate solution of Eq.(7), where $\mathbf{B}^*(\zeta^*)$ is the local minimum of $\mathcal{L}_{\zeta^*}(\mathbf{B})$ and ζ^* makes the function $\mathcal{L}_{\zeta^*}(\mathbf{B})$ become a concave function.

Proposition 3. *The optimal solution $\mathbf{B}^*(\zeta^*)$ of $\mathcal{L}$ in Eq.(8) is exact that of the original discrete problem of Eq.(4), where ζ^* makes the objective $\mathcal{L}_{\zeta^*}(\mathbf{B})$ become a concave function.*

Proof. See A.3 in supplementary file. □

D-subproblem

In this subproblem, we update $\mathbf{D}$ with fixed $\mathbf{B}$ by solving the following problem:

$$\mathcal{L}_\zeta(\mathbf{D}) = \begin{cases}(1-\zeta)\mathcal{L}(\mathbf{D}) + \zeta tr(\mathbf{D}^T\mathbf{D}), 1 \geq \zeta \geq 0;\\ (1+\zeta)\mathcal{L}(\mathbf{D}) + \zeta tr(\mathbf{D}^T\mathbf{D}), 0 \geq \zeta \geq -1,\end{cases}$$
$$s.t., \mathbf{D} \in \mathcal{P}_\mathbf{D}, \mathcal{P}_\mathbf{D} := \{\mathbf{D} : \mathbf{D} \in [-1, 1]^{f\times n}\}. \quad (11)$$

For the conciseness, we do not give the derivation of the updating rule which could be derived in a similar way just like in B-subproblem.

4.3 Algorithmic Analysis

The convergence of the proposed DLPR model is guaranteed by the following proposition.

Proposition 4. *(Convergence of DLPR model). The sequence $\{\mathbf{B}(\zeta), \mathbf{D}(\zeta)\}$ monotonically decreases the objective function $\mathcal{L}$ of Eq.(7); the objective function has a lower bound; the solution sequence $\{\mathbf{B}(\zeta), \mathbf{D}(\zeta)\}$ converges.*

Proof. See A.4 in supplementary file. □

5 Experiments

5.1 Experimental Settings

Datasets

Two real-world datasets are used in our experiments, namely CiteULike and Amazon, where the Amazon Book dataset is converted into implicit feedback by treating ratings of higher than each user's average score as "like" preference. In addition, we filter users who have less than 10 interactions due to extreme sparsity. Table 1 summarizes the filtered datasets.

Datasets	#Users	#Items	#Ratings	Density
CiteULike	5,139	16,980	200,866	0.230%
Amazon	31,058	33,193	1,019,442	0.099%

Table 1: Statistics of the datasets used in our experiments.

Baselines

We compare DLPR with four state-of-the-art HR methods:

- **DCF** [Zhang *et al.*, 2016] is a canonical one-stage HRex method whose objective function is rating prediction instead of personalized items ranking.
- **DPR** [Zhang *et al.*, 2017] is an HRim method which optimizes a surrogate AUC loss.
- **DRMF** [Zhang *et al.*, 2018a] is an HRim method which optimizes the quadratic upper bound of logistic loss based on pairwise preferences, and introduce self-paced learning to optimization procedure.
- **DMF** [Lian *et al.*, 2021] is an HRim method which optimizes quadratic upper bound of logistic loss.

To verify whether each component adopted by DLPR is useful or not, several variants of DLPR are included:

- **DLPR-MQ (Median Quantization)** conducts median quantization on the real-valued representations learned by LPR which is the real-valued version of DLPR.
- **DLPR-PL (Pairwise Loss)** replaces AP loss with pairwise loss. Remaining components are same as DLPR.
- **DLPR-PL-OL (One-stage Learning)** is a further degenerated version of DLPR-PL, which adopts one-stage method to learn hash codes by optimizing pairwise loss.

Evaluation Protocols

To evaluate the performance of our method, we adopt the leave-one-out evaluation, which has been widely used in literature [Zhang *et al.*, 2017; He *et al.*, 2017]. We exploit four widely-used metrics in ranking evaluation: Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), Recall and Mean Reciprocal Rank (MRR).

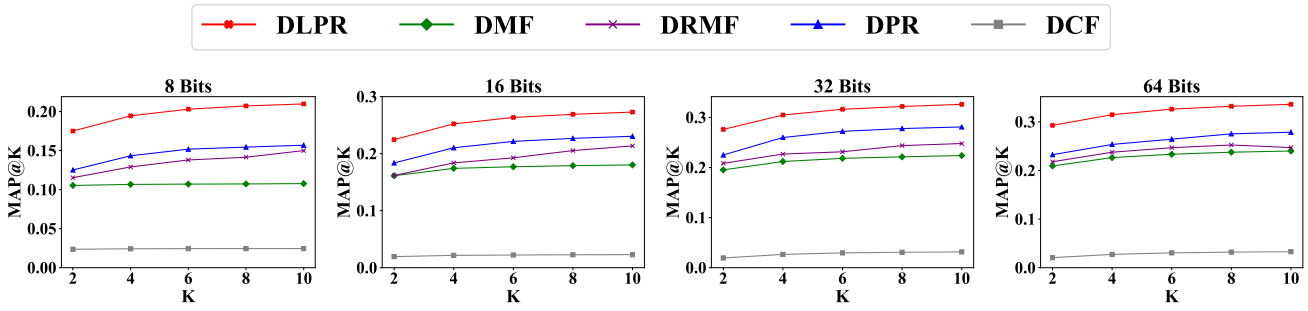
Parameter Settings

We employ grid search for parameters tuning and best results are reported. We test the temperature parameter τ in the range of $\{10^{-1}, 10^0, \cdots, 10^3\}$ and the optimization step d in the range of $\{0.01, 0.02, 0.05, 0.1, 0.2\}$. For baselines, we use the original implementation as released by the authors.

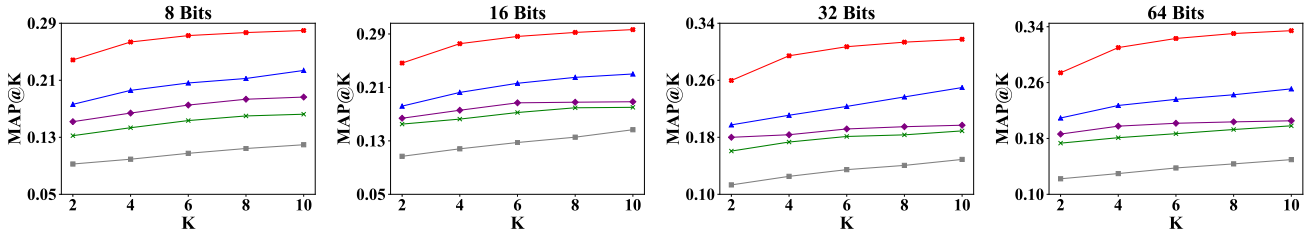
5.2 Comparison with State of the Arts

We first compare DLPR with the four baselines on the two datasets, and the corresponding quantitative results are shown in Figure 2 and Table 2. Specifically, several observations are drawn from the experimental results:

- Overall, listwise HRim method (DLPR) and pairwise HRim methods (DRMF and DPR) consistently outperform pointwise HR methods (DCF, DMF). Such an observation illustrates the importance of listwise and pairwise loss for binary codes learning in implicit scenarios. Between the two pointwise HR methods, DMF consistently outperforms DCF. That's because DMF learns



(a) CiteULike



(b) Amazon

Figure 2: Performance of our proposed DLPR compared with four state-of-the-art HR methods on CiteULike and Amazon datasets given different code length.

Methods	CiteULike			Amazon		
	NDCG@10	Recall@10	MRR	NDCG@10	Recall@10	MRR
DLPR	**0.33975**	**0.56042**	**0.33288**	**0.36042**	**0.61179**	**0.34986**
DMF	0.24090	0.40260	0.25168	0.22047	0.40366	0.23524
DRMF	0.24526	0.44442	0.26233	0.25193	0.42306	0.25379
DPR	0.26950	0.50730	0.28327	0.28131	0.50792	0.29555
DCF	0.03616	0.08620	0.05058	0.18192	0.32291	0.18973

Table 2: Performance of our proposed DLPR model with four state-of-the-art HR methods with NDCG@10, Recall@10 and MRR on CiteuLike and Amazon Datasets with '64-bits' codes.

hash codes by optimizing logistic loss which is closer to the ranking task than squared loss DCF optimizes. In addition, we find DCF shows worse recommendation performance on CiteULike, which is due to the fact that CiteULike is an implicit dataset while DCF learns hash codes by optimizing squared loss.

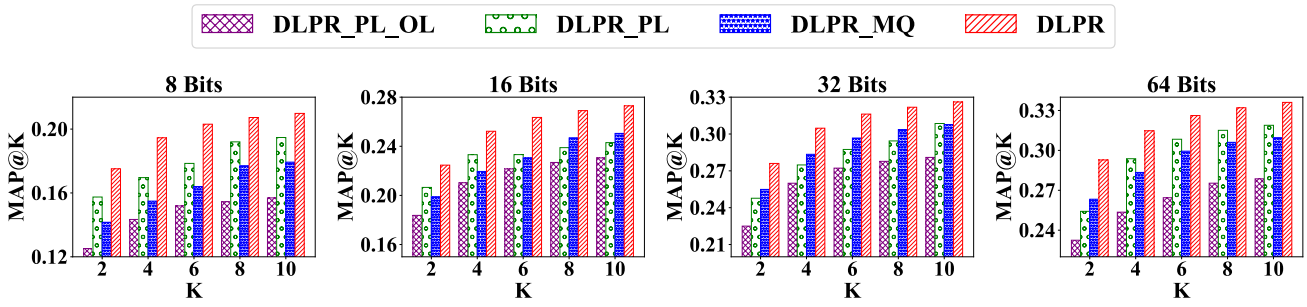
- DLPR demonstrates consistent improvements over state-of-the-art pairwise HRim methods (DRMF and DPR) on two datasets in terms of MAP@K. The performance improvements are attributed to the benefits of joint effect of the proposed smooth AP loss and the corresponding optimization algorithm. Meanwhile, DLPR also obtains higher MRR and Recall@10 than the two pairwise methods, which verifies the fact optimizing more information metric, like AP, can boost the performance of the less informative metrics, such as Recall and MRR [Li *et al.*, 2021]. In addition, our method achieves significant performance improvement on NDCG@10 since NDCG is similar with AP and optimizing AP is actually optimizing NDCG. By comparing DRMF and DPR, we find DPR shows better performance than DRMF since DPR directly optimizes surrogate AUC objective.

5.3 Ablation Experiments

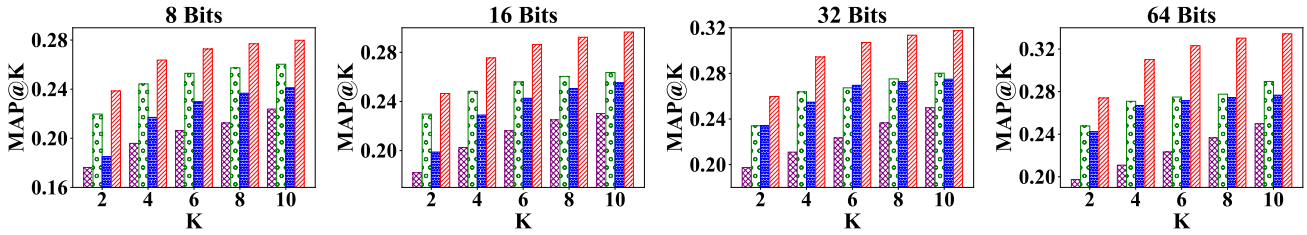
We further compare DLPR with its three degenerated variants (i.e., DLPR-PL, DLPR-MQ and DLPR-PL-OL) for demonstrating the effectiveness of key components adopted by DLPR. Figure 3 and Table 2 respectively show MAP@K, NDCG@10, Recall@10 and MRR of such four methods.

Effect of the Smooth AP Loss

Comparing DLPR with DLPR-PL, which are solved based on the same optimization algorithm, we find that DLPR significantly outperforms DLPR-PL. It verifies that the proposed AP loss is beneficial in binary codes learning. Moreover, it is surprising that DLPR-MQ achieves much better performance than DLPR-PL-OL. Note that DLPR-MQ performs crude median quantization to obtain hash codes, while DLPR-PL-OL does that via the one-stage learning method. It again confirms that the importance of the consistency between learning objective and evaluation criterion for recommendation.



(a) CiteULike



(b) Amazon

Figure 3: Performance of our proposed DLPR compared with DLPR-MQ, DLPR-PL and DLPR-PL-OL on CiteULike and Amazon datasets given different code length.

Methods	CiteULike			Amazon		
	NDCG@10	Recall@10	MRR	NDCG@10	Recall@10	MRR
DLPR	**0.33975**	**0.56042**	**0.33288**	**0.36042**	**0.61179**	**0.34986**
DLPR-MQ	0.30132	0.53894	0.29183	0.31486	0.54729	0.31055
DLPR-PL	0.31984	0.54882	0.31144	0.32768	0.56656	0.32222
DLPR-PL-OL	0.26950	0.50730	0.28327	0.28131	0.50792	0.29555

Table 3: Performance of our proposed DLPR model with DLPR-MQ, DLPR-PL and DLPR-PL-OL with NDCG@10, Recall@10 and MRR on CiteuLike and Amazon Datasets with '64-bits' codes.

Effect of the Alternating Optimization Strategy

Comparing DLPR-PL with DLPR-PL-OL, which are with the same objective, we observe that DLPR-PL consistently outperforms DLPR-PL-OL. It verifies the superiority of our proposed optimization strategy over the popular one-stage learning paradigm that is widely used in HR. In contrast with one-stage learning, the proposed optimization strategy can directly obtain the *discrete solution* in the *continuous vector space*, which is helpful to enhance the representation ability of DLPR. Besides, we also find a large performance gap between DLPR and DLPR-MQ, which further demonstrates the superiority of our proposed optimization strategy over the traditional two-stage learning technique.

We also conduct efficiency comparison between DLPR and its real-valued version to show the superiority of our method, whose details are shown in B.1 of the supplementary file.

6 Conclusion and Future Work

Motivated by the fact that current HRim schemes fail to align the learning objective with AP metric, we proposed DLPR that is the first personalized ranking method to optimize AP under discrete constraints. Specifically, DLPR directly learns binary codes in a relaxed continuous solution space, and we provide an analysis to verify that the optimal solution of the relaxed continuous optimization problem is same as that of the original discrete problem. Extensive experiments verify the superiority of DLPR over several competitive baselines.

To make use of the representation ability of deep neural networks, [Wang *et al.*, 2020] sought the approximate solution to discrete optimization problem within the deep learning framework. Towards this line, we will explore the discrete listwise deep recommendation models to further improve the performance of HRim techniques.

Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities (No. 2021YJS038), the National Nature Science Foundation of China (No. 62076021) and the Beijing Municipal Natural Science Foundation (No. 4202060).

References

[Brown *et al.*, 2020] Andrew Brown, Weidi Xie, Vicky Kalogeiton, and Andrew Zisserman. Smooth-ap: Smoothing the path towards large-scale image retrieval. In *ECCV*, volume 12354, pages 677–694, 2020.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198, 2016.

[Grover *et al.*, 2019] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *ICLR*, 2019.

[Hansen *et al.*, 2021] Christian Hansen, Casper Hansen, Jakob Grue Simonsen, and Christina Lioma. Projected hamming dissimilarity for bit-level importance coding in collaborative filtering. In *WWW*, pages 261–269, 2021.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, page 173–182, 2017.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Li *et al.*, 2021] Roger Zhe Li, Julián Urbano, and Alan Hanjalic. New insights into metric optimization for ranking-based recommendation. In *SIGIR*, pages 932–941, 2021.

[Lian *et al.*, 2021] Defu Lian, Xing Xie, and Enhong Chen. Discrete matrix factorization and extension for fast item recommendation. *IEEE Trans. Knowl. Data Eng.*, 33(5):1919–1933, 2021.

[Liu and Qiao, 2014] Zhiyong Liu and Hong Qiao. GNCCP - graduated nonconvexityand concavity procedure. *IEEE T. Pattern Anal.*, 36(6):1258–1267, 2014.

[Liu *et al.*, 2019] Chenghao Liu, Xin Wang, Tao Lu, Wenwu Zhu, Jianling Sun, and Steven Hoi. Discrete social recommendation. In *AAAI*, 2019.

[Liu *et al.*, 2021] Chenghao Liu, Tao Lu, Zhiyong Cheng, Xin Wang, Jianling Sun, and Steven C. H. Hoi. Discrete listwise collaborative filtering for fast recommendation. In *SDM*, pages 46–54, 2021.

[Luo *et al.*, 2021] Fangyuan Luo, Jun Wu, and Haishuai Wang. Semi-discrete social recommendation (student abstract). In *AAAI*, pages 15835–15836, 2021.

[Marguerite Frank, 1956] Philip Wolfe Marguerite Frank. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Rockafellar, 1997] R.T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1997.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015.

[Wang *et al.*, 2012] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012.

[Wang *et al.*, 2016] J Wang, W Liu, S Kumar, and S Chang. Learning to hash for indexing big data - a survey. *Proc. IEEE*, 104(1):34–57, 2016.

[Wang *et al.*, 2018a] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *IEEE T. Pattern Anal.*, 40(4):769–790, 2018.

[Wang *et al.*, 2018b] Tao Wang, Haibin Ling, Congyan Lang, and Songhe Feng. Graph matching with adaptive and branching path following. *IEEE T. Pattern Anal.*, 40(12):2853–2867, 2018.

[Wang *et al.*, 2020] Tao Wang, He Liu, Yidong Li, Yi Jin, Xiaohui Hou, and Haibin Ling. Learning combinatorial solver for graph matching. In *CVPR*, pages 7565–7574, 2020.

[Wu *et al.*, 2020] Jun Wu, Fangyuan Luo, Yujia Zhang, and Haishuai Wang. Semi-discrete matrix factorization. *IEEE Intell. Syst.*, 35(5):73–83, 2020.

[Zhang *et al.*, 2014] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. Preference preserving hashing for efficient recommendation. In *SIGIR*, pages 183–192, 2014.

[Zhang *et al.*, 2016] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat Seng Chua. Discrete collaborative filtering. In *SIGIR*, pages 325–334, 2016.

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

[Zhang *et al.*, 2018a] Yan Zhang, Haoyu Wang, Defu Lian, Ivor W. Tsang, Hongzhi Yin, and Guowu Yang. Discrete ranking-based matrix factorization with self-paced learning. In *SIGKDD*, pages 2758–2767, 2018.

[Zhang *et al.*, 2018b] Yujia Zhang, Jun Wu, and Haishuai Wang. Binary collaborative filtering ensemble. In *PRICAI*, pages 993–1004, 2018.

[Zhang *et al.*, 2019] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1):5:1–5:38, 2019.

[Zhang *et al.*, 2020] Yan Zhang, Ivor W. Tsang, Hongzhi Yin, Guowu Yang, Defu Lian, and Jingjing Li. Deep pairwise hashing for cold-start recommendation. *CoRR*, abs/2011.00944, 2020.

[Zhou and Zha, 2012] Ke Zhou and Hongyuan Zha. Learning binary codes for collaborative filtering. In *SIGKDD*, pages 498–506, 2012.