

Discrete Scale-invariant Metric Learning for Efficient Collaborative Filtering

YAN ZHANG*, Faculty of Science and Technology, Charles Darwin University, Darwin, Australia

LI DENG, University of Electronic Science and Technology of China, Chengdu, China

LIXIN DUAN, Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen, China

SAMI AZAM, Faculty of Science and Technology, Charles Darwin University, Darwin, Australia

Metric learning has attracted extensive interest for its ability to provide personalized recommendations based on the importance of observed user-item interactions. Current metric learning methods aim to push negative items away from the corresponding users and positive items by an absolute geometrical distance margin. However, items may come from imbalanced categories with different intra-class variations. Thus, the absolute distance margin may not be ideal for estimating the difference between user preferences over imbalanced items. To this end, we propose a new method, named discrete scale-invariant metric learning (DSIML), by adding binary constraints to users and items, which maps users and items into binary codes of a shared Hamming subspace to speed up the online recommendation. Specifically, we firstly propose a scale-invariant margin based on angles at the negative item points in the shared Hamming subspace. Then, we derive a scale-invariant triple hinge loss based on the margin. To capture more preference difference information, we integrate a pairwise ranking loss into the scale-invariant loss in the proposed model. Due to the difficulty of directly optimizing the mixed integer optimization problem formulated with *log-sum-exp* functions, we seek to optimize its variational quadratic upper bound and learn hash codes with an alternating optimization strategy. Experiments on benchmark datasets clearly show that our proposed method is superior to competitive metric learning and hashing-based baselines for recommender systems.

CCS Concepts: • Information systems → Recommender systems; Learning to rank;

Additional Key Words and Phrases: Recommender system, Deep Learning, Hash code, Cold-start

ACM Reference Format:

Yan Zhang, Li Deng, Lixin Duan, and Sami Azam. 2025. Discrete Scale-invariant Metric Learning for Efficient Collaborative Filtering. 1, 1 (December 2025), 16 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 INTRODUCTION

Personalized recommender systems, as one of the most critical and effective approaches for alleviating information overload, have been successfully applied in various areas including online e-commerce websites: Amazon, Netflix, Yelp, etc., online educational systems, and even online

*Corresponding author.

Authors' addresses: Yan Zhang, yan.zhang@cdu.edu.au, Faculty of Science and Technology, Charles Darwin University, Darwin, Australia; Li Deng, 136995963@qq.com, University of Electronic Science and Technology of China, Chengdu, China; Lixin Duan, lxduan@uestc.edu.cn, Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen, China; Sami Azam, sami.azam@cdu.edu.au, Faculty of Science and Technology, Charles Darwin University, Darwin, Australia.

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 the author(s) 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.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.

XXXX-XXXX/2025/12-ART \$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

health-care systems. Personalized recommender systems often provide top- k ranked list of items for a specific user to assist users to target their interests. As implicit feedback information is more common than explicit rating data in the real world applications, we focus on recommendations based on the implicit feedback data in this paper.

Efforts have been made to improve the accuracy of personalized recommendations by estimating the preference difference between positive and negative items based on implicit feedback (e.g., clicks, browse recodes, rating records, etc.). These efforts are two folds: (1) ranking-based recommender systems consisting of pairwise learning to rank (LTR) [6, 11, 15–17, 28] and listwise LTR methods [23, 27, 29]; and (2) metric learning recommender systems [5, 10, 14].

In pairwise LTR methods, one representative is Bayesian personalized ranking (BPR) [17] and its variant [16] based on a non-uniform item sampler, which is proposed to speed up the convergence under the circumstance that item popularity has a tailed distribution. To achieve better ranking performances, nBRP-MF [11] formulates ranking tasks of recommendations based on the Rank-Biased Precision(BRP) [15]. It evaluates that both pairwise and listwise ranking-based recommendation models can achieve better performances for active users with more of the relevant interactions compared to less active users. To address the ranking problem for binary relevance datasets, collaborative less-is-more filtering (CLiMF) [19] is proposed by directly maximizing the Mean Reciprocal Rank (MRR). To achieve efficient recommendations with a simpler student model, a ranking distillation method [24, 33] is proposed for learning to rank problems by training a teacher-student model.

In listwise LTR methods, one representative is CoFi-Rank [27], which optimizes a listwise ranking-based objective motivated by the idea of maximum margin matrix factorization (MMMF) [23]. Then, Yue et. al [20] proposes the ListRank-MF for collaborative filtering that combines a listwise LTR objective with matrix factorization. A ranked list of items is obtained by minimizing a loss function that represents the uncertainty between training lists and output lists produced by a MF ranking model. Following the memory-based collaborative filtering (CF) framework, Shanshan et al [7] proposes the ListCF that predicts a total order of items for each user based on similar users' probability distributions over permutations of the items. One important advantage of ListCF lies in its ability of reducing the computational complexity of the training and prediction procedures while achieving the same or better ranking performances as compared to previous ranking-oriented memory-based CF algorithms. To relieve the expensive computational cost and provide a theoretical analysis for the listwise ranking model, SQL-Rank [29] is proposed by taking listwise collaborative ranking as maximum likelihood under a permutation model which applies probability mass to permutations based on a low-rank latent rating matrix.

Despite that, ranking-based recommendations are powerful for learning the user preference difference by learning discriminative latent factors for users and items by pairwise ranking losses and listwise ranking objectives. However, they suffer from a crucial drawback stemming from MF-based methods that violate the triangle inequality according to [21], which leads to the failure of capturing more preference information [5]. To address the limitations of MF-based methods, metric learning approaches have been studied in recommendations [5, 10, 14, 30, 34]. These methods project users and items into a low-dimension metric space, where the distance estimates the user preference for the item. Nevertheless, the existing metric learning method has the limitation of assuming all users and items share the same fixed margin m , which is inappropriate for personalized recommendations.

Accordingly, it is challenging for existing metric learning methods to learn the user preference difference when facing imbalance data shown in Figure 1. It shows two classes of books on Amazon: Economics and Tourism. Economics is the majority class, which is much more diverse than the minority class Tourism books, and the scale of Economics books is much larger than Tourism

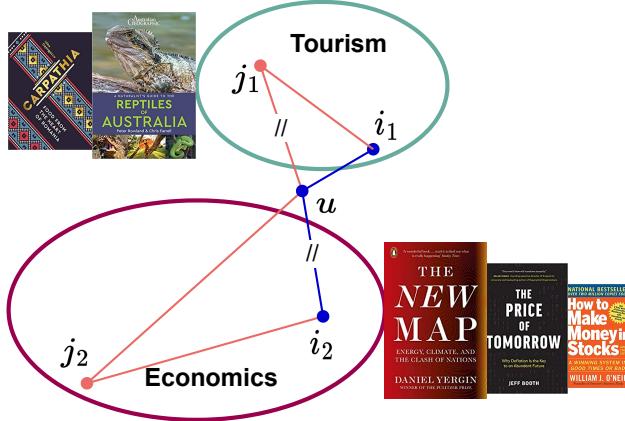


Fig. 1. An illustration of the challenge brought by imbalanced items. User u has positive items i_1, i_2 and negative items j_1, j_2 from two imbalanced classes of books: the majority Economics and the minority Tourism. An absolute distance margin is difficult to measure the user preference difference between i_2 and j_1 when $d(u, j_1) = d(u, i_2)$.

books. Therefore, it's not applicable to fix the same margin for the imbalanced item dataset. For user u , the positive (interested) items are i_1, i_2 , the negative (not interested) items are j_1, j_2 . Existing metric learning methods push the negative item j_1 away from user u with an absolute geometrical distance margin m when training on the triplets (u, i_1, j_1) and (u, i_2, j_2) ; meanwhile, it will push the positive item i_2 away from user u because $d(u, j_1) = d(u, i_2)$, which hinders to learn the user preference difference between items j_1 and i_2 . So a geometrical distance margin is not feasible to estimate the user preference difference to items from imbalanced classes.

2 THE PROPOSED MODEL

In this section, we first introduce the notations used in this Chapter. Next, we intuitively analyze the relation of a user, her/his positive item, and the negative item of a triplet in Hamming space. Then, we introduce the model formulation of DSIML in Section 2.2. This section first proposes a scale-invariant margin for metric learning in Hamming space. To capture more preference information, we integrate a pairwise ranking loss with the scale-invariant loss in the proposed model DSIML.

2.1 Notations

Let users and items be denoted as $\mathcal{U} = \{1, \dots, n\}$ and $\mathcal{I} = \{1, \dots, m\}$ respectively. The implicit feedback \mathcal{S} is defined as a subset of the cartesian product of \mathcal{U} and \mathcal{I} : $\mathcal{S} \subseteq \mathcal{U} \times \mathcal{I}$, where $s_{ui} = 1$ denotes user u has an interaction/rating with item i ; otherwise, $s_{ui} = 0$. The task of personalized top- k recommendation is providing a personalized ranking list of k items from all items in \mathcal{I} for a specific user. The proposed method aims to provide personalized recommendations with metric learning in Hamming space by projecting users and items into d -dimension binary codes. I define positive items of user i as $\mathcal{I}_u^+ = \{i \in \mathcal{I} : (u, i) \in \mathcal{S}\} \subset \mathcal{I}$; and negative items as the remaining items $\mathcal{I}_u^- = \mathcal{I} \setminus \mathcal{I}_u^+ \subset \mathcal{I}$. Similarly, we define positive users who rated item i as $\mathcal{U}_i^+ = \{u \in \mathcal{U} : (u, i) \in \mathcal{S}\} \subset \mathcal{U}$; and negative users as other users $\mathcal{U}_i^- = \mathcal{U} \setminus \mathcal{U}_i^+ \subset \mathcal{U}$.

2.2 Model Formulation

Suppose that we are given a set of training triplets $\Omega = \{(u, i, j) : u \in \mathcal{U}, i \in \mathcal{I}_u^+, j \in \mathcal{I}_u^-\}$. In existing metric learning methods, such as CML [5], it aims to push negative item j away from user u by a distance margin $m > 0$ compared to the positive item i by minimizing the following relaxed surrogate hinge loss,

$$\mathcal{L}_{\text{triplet}} = [d^2(\mathbf{b}_u, \mathbf{d}_i) - d^2(\mathbf{b}_u, \mathbf{d}_j) + m]_+. \quad (1)$$

where $[\cdot]_+ = \max\{0, \cdot\}$ represents the hinge function. \mathbf{b}_u denotes the representation of user u . $\mathbf{d}_i, \mathbf{d}_j$ represent vectors of positive item i and negative item j , respectively.

As analyzed in SML [10] and PALAM [14], CML only considers user-item relations and ignores the item-item relations, so it probably drags the negative item towards the positive item, which contradicts the objective of metric learning. Hence, SML and PALAM are proposed by considering user-item and item-item relations.

However, SML and PALAM cannot deal with the case in Figure 1 because the absolute distance margin cannot be used to measure the user preference difference for items from imbalanced categories. In this paper, we design a scale-invariant margin to deal with the imbalanced problem in Hamming space. Suppose that $\mathbf{b}_u \in \mathbf{B}$ denotes the hash code of user u . $\mathbf{d}_i \in \mathbf{D}$ denotes the hash code of item j . The distance between user u and item j in Hamming space is denoted as $\text{Dis}_H(\mathbf{b}_u, \mathbf{d}_i)$. In the triplet $(u, i, j) \in \Omega$, if $\text{Dis}_H(\mathbf{b}_u, \mathbf{d}_i) < \text{Dis}_H(\mathbf{b}_u, \mathbf{d}_j)$, then user u is more interested in item i than item j . The Hamming distance Dis_H can be denoted by the inner product of two hash codes,

$$\begin{aligned} \text{Dis}_H(\mathbf{b}_u, \mathbf{d}_i) &= \sum_{k=1}^d \mathbb{I}(b_{uk} \neq d_{ik}) \\ &= \frac{1}{2}(d - \mathbf{b}_u^T \mathbf{d}_i) \\ &= \frac{1}{2}(d - \|\mathbf{b}_u\| \cdot \|\mathbf{d}_i\| \cos \theta_{ui}), \end{aligned} \quad (2)$$

where d is the dimension of hash codes \mathbf{b}_u and \mathbf{d}_i . θ_{ui} denotes the angle formed by the two binary vectors. From the above equation, we derive that the Hamming distance is consistent with the cosine value of the angle, because the norm of d -dimension hash code is constant \sqrt{d} . Consequently, a smaller angle leads to similar hash codes, which motivates us to find a scale-invariant margin based on the angles of hash codes.

As illustrated in Figure 2, the objective (1) of CML aims to minimize the edge \overline{ui} and push the negative sample j away from u with a geometrical margin m , that enforces $\|\overline{ui}\| + m \leq \|\overline{uj}\|$, which may drag the negative item j towards the positive item i . To avoid this, we penalize a longer edge \overline{ij} compared to the edge \overline{ui} . According to the cosine rule, it can be proved that the angle surrounded by the longer edge \overline{ui} and \overline{ij} should be the smallest one in the triangle $\triangle uij$ [26]. As the summarization of three angles in the triangle equals 180° , so the angle $\angle uji$ should be smaller than 60° , i.e., $\angle uji \leq 60^\circ$. Since items may belong to different categories with different intra-class variations, we aim to find an angle margin β imposed on the angle at the negative item point j instead of searching for an absolute distance margin m .

Given user u , the positive item i , and the negative item j , the proposed objective is to minimize $\angle uji$. If we decrease the angle $\angle uji$, the negative j may be pulled towards the positive i , which violates the proposed objective. So we try to find an alternative way to push the negative item j away from both the user and the positive item. The intuition is pushing negative sample j away from a local distribution formed by user u and the positive item i . Approximately, we assume the local distribution is the circle area built by the diameter \overline{ui} . Thus when we move the user u to the center o and move the positive item i to the item i' . $\|\overline{oi'}\| = \|\overline{oi}\| = \frac{1}{2}\|\overline{ui}\|$, and $\overline{oi'}$ is orthogonal to

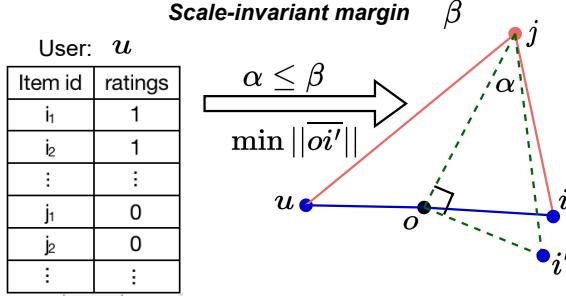


Fig. 2. The illustration of deriving scale-invariant margin. If I directly reduce the angle $\angle uij$, the negative point j may be dragged towards the positive point i , which violates the objective. Alternatively, we reduce α in the auxiliary triangle $\triangle o i' j$ with the intuition of pushing negative items away from an approximated local distribution formed by u and i . Hence, we impose a scale-invariant margin β on α , i.e., $\alpha \leq \beta$.

\overline{oj} . Given user u and positive item i , if we reduce the angle α , the negative item j will be pushed away from all points in the local distribution, including u, i, o , and i' . This motivates us to define an angle margin β on the angle α , and the proposed objective is

$$\alpha \leq \beta. \quad (3)$$

In the triangle $\triangle o i' j$,

$$\tan \alpha = \frac{||\overline{oi'}||}{||\overline{oj}||} \leq \tan \beta. \quad (4)$$

In the training procedure on the triplet $\{u, i, j\}$, we aim to update hash codes $\mathbf{b}_u, \mathbf{d}_i$, and \mathbf{d}_j by minimizing the following hinge loss:

$$\mathcal{L}_{scale} = [||\overline{oi'}||^2 - \gamma^2 ||\overline{oj}||^2]_+, \quad (5)$$

where $\gamma = \tan \beta$, $||\overline{oi'}||^2 = ||\overline{oi}||^2 = \frac{1}{4}(\mathbf{b}_u - \mathbf{d}_i)^2$, $||\overline{oj}||^2 = ||\mathbf{d}_j - \frac{1}{2}(\mathbf{b}_u + \mathbf{d}_i)||^2$.

If $\alpha \leq \beta$, i.e., $||\overline{oi'}||^2 \leq \gamma^2 ||\overline{oj}||^2$, we do not update hash codes $\mathbf{b}_u, \mathbf{d}_i$, and \mathbf{d}_j ; otherwise, we update them by minimizing the following scale-invariant margin loss:

$$\mathcal{L}_{scale} = ||\mathbf{b}_u - \mathbf{d}_i||^2 - \gamma^2 ||2\mathbf{d}_j - (\mathbf{b}_u + \mathbf{d}_i)||^2. \quad (6)$$

To better understand the effect of optimizing the scale-invariant margin loss, we investigate sub-problems regarding \mathbf{b}_u , \mathbf{d}_i and \mathbf{d}_j , respectively. By dropping terms irrelevant to \mathbf{b}_u when \mathbf{d}_i and \mathbf{d}_j are fixed, the sub-problem of \mathbf{b}_u is as follows,

$$\mathcal{L}_{scale-u} = -(1 + \gamma^2) \mathbf{b}_u^T \mathbf{d}_i + 2\gamma^2 \mathbf{b}_u^T \mathbf{d}_j, \quad (7)$$

we derive that the updates rule of \mathbf{b}_u is dependent on all samples in the triplet $(u, i, j) \in \Omega$. Similarly, we derive the sub-problem of \mathbf{d}_i and \mathbf{d}_j as follows,

$$\mathcal{L}_{scale-i} = -(1 + \gamma^2) \mathbf{b}_u^T \mathbf{d}_i + 2\gamma^2 \mathbf{d}_i^T \mathbf{d}_j, \quad (8)$$

$$\mathcal{L}_{scale-j} = 2\gamma^2 (\mathbf{b}_u + \mathbf{d}_i)^T \mathbf{d}_j. \quad (9)$$

Compared to CML, where each sample in a triplet is updated with only two points [10], which may easily stick into a locally optimal solution. In the proposed scale-invariant metric learning, each sample u, i , and j is updated by all three points. So it can achieve a better solution than CML.

Compared to SML and PALAM, the proposed scale-invariant margin is enforced on the angle at the negative point. The absolute distance margin of SML or PALAM varies with items from classes with different intra-class variations. Instead, the proposed margin does not change with different variations. So the margin is invariant for items with imbalanced classes.

However, the hinge loss (5) is a non-smooth objective when considering more than one negative point. So we convert the hinge loss to its smooth upper bound inspired by the recent work [22]. Because users and items are of constant length, i.e., $\|\mathbf{b}_u\| = \|\mathbf{d}_i\| = \|\mathbf{d}_j\| = \sqrt{d}$ in Equation(5). After dropping constant terms, Equation (5) is approximated by the following smooth *log-sum-up* upper bound,

$$\mathcal{L}_{scale} = \sum_{(u,i,j) \in \Omega} \log(1 + \exp(2\gamma^2(\mathbf{b}_u^T \mathbf{d}_i + \mathbf{d}_i^T \mathbf{d}_j) - (1 + \gamma^2)\mathbf{b}_u^T \mathbf{d}_i)), \quad (10)$$

To capture more preference information from triplets $(u, i, j) \in \Omega$, we add the pairwise ranking loss to the objective inspired by BPR [17]. In Hamming space, we assume that the rating score is predicted by $\hat{r}_{ui} = 1 - \frac{1}{d}\text{Dis}_H(\mathbf{b}_u, \mathbf{d}_i) = \frac{1}{2} + \frac{1}{2d}\mathbf{b}_u^T \mathbf{d}_i$. So the pairwise ranking loss is formulated as:

$$\mathcal{L}_{pair} = \sum_{(u,i,j) \in \Omega} \mathbb{I}(\hat{r}_{ui} - \hat{r}_{uj}), \quad (11)$$

where $\mathbb{I}(\cdot)$ is a delta function, which returns 1 if the input is true and 0 otherwise. Optimizing the above objective often directly leads to an NP-hard problem [1]. A solvable solution is to minimize some pairwise surrogate losses as follows,

$$\mathcal{L}_{pair} = \sum_{(u,i,j) \in \Omega} \ell(\hat{r}_{ui} - \hat{r}_{uj}) \quad (12)$$

where $\ell : \mathbb{R} \rightarrow \mathbb{R}^+$ is the surrogate *log-sum-exp* function [35] as follows

$$\mathcal{L}_{pair} = \sum_{(u,i,j) \in \Omega} \log(1 + \exp(\frac{\mathbf{b}_u^T \mathbf{d}_j - \mathbf{b}_u^T \mathbf{d}_i}{2d})). \quad (13)$$

Optimizing the above objective can capture more user preference difference information between positive and negative items.

By combining the pairwise ranking loss (13) and the scale-invariant loss (10), we formulate DSIML as the following objective,

$$\min_{\mathbf{B}, \mathbf{D}} \mathcal{L}_{dsiml}(\mathbf{B}, \mathbf{D}) = \mathcal{L}_{pair} + \lambda \mathcal{L}_{scale}, \quad (14)$$

where λ is a trade-off hyper-parameter that weights the importance between the pairwise loss and the scale-invariant loss. In the experiments, we test the effect of λ on recommendation performances. Its continuous version of DSIML, referred to as SIML has the same objective of dropping the binary constraints.

3 MODEL OPTIMIZATION

Due to the intractability of solving the objective (14) with binary constraints imposed on \mathbf{B} and \mathbf{D} , it's hard to find a closed form updates rule. Inspired by the new discrete optimization method [36], we seek to minimize a variational quadratic upper bound of the objective of DSIML in Equation (14) based on *JJ bound* [8]. The upper bound of *log-sum-exp* function is as follows

$$\log(1 + e^t) \leq \pi(\xi)(t^2 - \xi^2) + \frac{1}{2}(t - \xi) + \log(1 + e^\xi), \quad (15)$$

where $\pi(\xi) = \frac{1}{2\xi} (\sigma(\xi) - \frac{1}{2})$, $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function. we call ξ as the variational parameter. It's evident that the upper bound is tight at $\xi = t$. By denoting $\hat{\ell}(t, \xi) = \pi(\xi)(t^2 - \xi^2) + \frac{1}{2}(t - \xi) + \log(1 + e^\xi)$, we derive the following upper bound of Equation (14),

$$\begin{aligned} \min_{\mathbf{B}, \mathbf{D}, \Phi, \mathcal{H}} \hat{\mathcal{L}}_{dsiml} &= \sum_{(u, i, j) \in \Omega} (\hat{\ell}(x_{uij}, \varphi_{uij}) + \lambda \hat{\ell}(y_{uij}, \eta_{uij})), \\ \text{s.t. } \mathbf{B} \in \{\pm 1\}^{n \times d}, \mathbf{D} \in \{\pm 1\}^{m \times d}, \Phi \in \mathbb{R}^{|\Omega|}, \mathcal{H} \in \mathbb{R}^{|\Omega|}. \end{aligned} \quad (16)$$

where $x_{uij} = \frac{1}{2d} (\mathbf{b}_u^T \mathbf{d}_j - \mathbf{b}_u^T \mathbf{d}_i)$ is from Equation (13), and $y_{uij} = 2\gamma^2 (\mathbf{b}_u^T \mathbf{d}_j + \mathbf{d}_i^T \mathbf{d}_j) - (1 + \gamma^2) \mathbf{b}_u^T \mathbf{d}_i$ is from Equation (10). $\varphi_{uij} \in \Phi$ and $\eta_{uij} \in \mathcal{H}$ are the variational parameters.

To optimize the above mixed integer problem, we adopt an alternating optimization method, which is commonly applied in solving mixed integer problems [32, 35].

Updating users' hash codes \mathbf{B} . Given \mathbf{D} , Φ , and \mathcal{H} , we update \mathbf{b}_u of each user u in parallel. By dropping irrelevant items to \mathbf{b}_u , the sub-problem is a binary quadratic problem (BQP) with respect to \mathbf{b}_u ,

$$\min_{\mathbf{b}_u \in \{\pm 1\}^d} \mathbf{b}_u^T \mathbf{A}_u \mathbf{b}_u + \mathbf{c}_u^T \mathbf{b}_u, \quad (17)$$

where $\mathbf{A}_u = \mathbf{A}_{xu} + 4\lambda d^2 \mathbf{A}_{yu}$, $\mathbf{c}_u = \mathbf{c}_{xu} + 4\lambda d^2 \mathbf{c}_{yu}$. \mathbf{A}_{xu} and \mathbf{c}_{xu} come from the loss $\hat{\ell}(x_{uij}, \varphi_{uij})$, and \mathbf{A}_{yu} and \mathbf{c}_{yu} come from the loss $\hat{\ell}(y_{uij}, \eta_{uij})$, and

$$\left\{ \begin{array}{l} \mathbf{A}_{xu} = \sum_{(i,j)} (\mathbf{d}_i \mathbf{d}_j^T \pi(\varphi_{uij}) + \mathbf{d}_j \mathbf{d}_i^T \pi(\varphi_{uij}) - 2 \mathbf{d}_j \mathbf{d}_i^T \pi(\varphi_{uij})); \\ \mathbf{A}_{yu} = 4\gamma^4 \sum_{(i,j)} (4(1 + \gamma^2)^2 \mathbf{d}_i \mathbf{d}_i^T \pi(\eta_{uij}) + \mathbf{d}_j \mathbf{d}_j^T \pi(\eta_{uij}) - (1 + \gamma^2) \mathbf{d}_j \mathbf{d}_i^T \pi(\eta_{uij})); \\ \mathbf{c}_{xu} = \sum_{(i,j)} d_j (\mathbf{d}_j - \mathbf{d}_i); \\ \mathbf{c}_{yu} = \sum_{(i,j)} (4\gamma^4 (\mathbf{d}_i^T \mathbf{d}_j) \mathbf{d}_j \pi(\eta_{uij}) - 8\gamma^4 (1 + \gamma^2) (\mathbf{d}_i^T \mathbf{d}_j) \mathbf{d}_i + 2\gamma^2 \mathbf{d}_j - (1 + \gamma^2) \mathbf{d}_i). \end{array} \right. \quad (18)$$

Previous studies mainly apply the Discrete coordinate descent(DCD) [18] method to learn hash codes in a bitwise way. As analyzed in DRMF [36], the discrete coordinate descent method easily falls into local optima and is very sensitive to the initialization of parameters. In contrast to discrete coordinate descent, we update the entire hash code \mathbf{b}_u in each iteration by directly optimizing the BQP problem with the BQP solver¹. In this paper, we choose the Xpress optimizer to optimize the above BQP problem by transferring hash codes $\mathbf{b}_u \in \{\pm 1\}^d$ to boolean vectors $\frac{1}{2}(\mathbf{b}_u + \mathbf{1}_d) \in \{0, 1\}^d$.

Updating items' hash codes \mathbf{D} . Similarly, given \mathbf{B} , Φ , and \mathcal{H} , we update \mathbf{d}_j of each item j in parallel. By dropping irrelevant items to \mathbf{d}_j , the sub-problem is a binary quadratic problem (BQP) with respect to \mathbf{d}_j ,

$$\min_{\mathbf{d}_i \in \{\pm 1\}^d} \mathbf{d}_i^T \mathbf{A}_i \mathbf{d}_i + \mathbf{c}_i^T \mathbf{d}_i, \quad (19)$$

¹<https://www.cvxpy.org>

where $\mathbf{A}_i = \mathbf{A}_{xi} + 4\lambda d^2 \mathbf{A}_{yi}$, $\mathbf{c}_i = \mathbf{c}_{xi} + 4\lambda d^2 \mathbf{c}_{yi}$. Similarly, \mathbf{A}_{xu} and \mathbf{c}_{xu} come from the loss $\hat{\ell}(x_{uij}, \varphi_{uij})$, and \mathbf{A}_{yu} and \mathbf{c}_{yu} come from the loss $\hat{\ell}(y_{uij}, \eta_{uij})$, and

$$\begin{cases} \mathbf{A}_{xu} = \sum_{(u,j)} \mathbf{b}_u \mathbf{b}_u^T \pi(\varphi_{uij}); \\ \mathbf{A}_{yj} = \sum_{(u,j)} (4\gamma^4 \mathbf{d}_j \mathbf{d}_j^T \pi(\eta_{uij}) + (1+\gamma^2)^2 \mathbf{b}_u \mathbf{b}_u^T \pi(\eta_{uij}) - 4\gamma^2(1+\gamma^2) \mathbf{d}_j \mathbf{b}_u^T \pi(\eta_{uij})); \\ \mathbf{c}_{xi} = \sum_{(u,i)} (-2(\mathbf{b}_u^T \mathbf{d}_j) \mathbf{b}_u \pi(\varphi_{uij}) - d \mathbf{b}_u); \\ \mathbf{c}_{yu} = \sum_{(u,j)} (4\gamma^4 (\mathbf{b}_u^T \mathbf{d}_j) \mathbf{d}_j \pi(\eta_{uij}) - 2\gamma^2(1+\gamma^2) (\mathbf{b}_u^T \mathbf{d}_j) \mathbf{b}_u \pi(\eta_{uij}) + 2\gamma^2 \mathbf{d}_j - (1+\gamma^2) \mathbf{b}_u). \end{cases} \quad (20)$$

Similarly, we optimize the above BQP with the assist of Xpress solver by transferring hash codes $\mathbf{d}_i \in \{\pm 1\}^d$ to boolean vectors $\frac{1}{2}(\mathbf{d}_i + \mathbf{1}_d) \in \{0, 1\}^d$.

Updating variational parameters Φ . Given \mathbf{B} , \mathbf{D} , and \mathcal{H} , we obtain the following sub-objective of φ_{uij} by discarding irrespective terms of φ_{uij} ,

$$\min_{\varphi_{uij} \in \mathbb{R}} \pi(\varphi_{uij})(x_{uij}^2 - \varphi_{uij}^2) + \frac{1}{2}(x_{uij} - \varphi_{uij}) + \log(1 + e^{\varphi_{uij}}) \quad (21)$$

The sub-problem (21) is convex since its second derivative is greater than zero. Hence, the optimal value is achieved when the derivation of Equation (21) equals zero. For each user, the derivation of Equation (21) equals zero at

$$\varphi_{uij} = x_{uij} = \frac{1}{2d}(\mathbf{b}_u^T \mathbf{d}_j - \mathbf{b}_u^T \mathbf{d}_i) \quad (22)$$

In practical, we don't need store $\{\varphi_{uij} : \forall (u, i, j) \in \Omega\}$ in the training procedure. we only need to store all expressions x_{uij} and then update φ_{uij} . we do also not need to store $\pi(\varphi_{uij})$, since it can be computed on-the-fly in constant time when we cache these x_{uij} .

Updating variational parameters \mathcal{H} . Similarly, given \mathbf{B} , \mathbf{D} , and Φ , we derive the following sub-problem of η_{uij} by dropping irrelevant terms of η_{uij} in Equation(21),

$$\min_{\eta_{uij} \in \mathbb{R}} \pi(\eta_{uij})(y_{uij}^2 - \eta_{uij}^2) + \frac{1}{2}(y_{uij} - \eta_{uij}) + \log(1 + e^{\eta_{uij}}) \quad (23)$$

Similarly, the sub-objective (23) is convex and the optimal value is achieved when the derivation of Equation (23) equals to zero. For each user, the derivation of Equation (23) equals to zero at

$$\eta_{uij} = y_{uij} = 2\gamma^2(\mathbf{b}_u^T \mathbf{d}_i + \mathbf{d}_i^T \mathbf{d}_j) - (1 + \gamma^2)\mathbf{b}_u^T \mathbf{d}_i \quad (24)$$

Similarly, we don't need store η_{uij} and $\pi(\eta_{uij})$, and we only need to store all expressions y_{uij} and then update η_{uij} and $\pi(\eta_{uij})$ with these y_{uij} . Finally, we summarize the optimization steps in Algorithm 1.

In the training stage, we optimize the objective (14) by a mini-batch containing multiple triplets from the same users. Specifically, we divide triplets, w.r.t users. Suppose the batch size is 16, we train the proposed model on all triplets of 16 users. In training, we optimize user by user, i.e., we train the proposed model in the way of N-pair sampling [22]. It considers an $(N+1)$ -tuple of training examples $\{u, i, j_1, j_2, \dots, j_{N-1}\}$ by sampling $N-1$ negative samples for each positive feedback $s_{ij} = 1$. As evaluated in NCF [4], the recommender system can achieve good performance when we randomly sample four to eight negative samples for each positive feedback. So in this Chapter, we sample five negative samples for each positive item to train the proposed model.

Algorithm 1: DSIML

Input: User-item implicit feedback \mathcal{S} , dimension $d = 20$, $\gamma = 1$, $\lambda = 1$.
Output: Hash codes of users and items \mathbf{B} and \mathbf{D} .

1 **Initialize:** Firstly we learn continuous vectors \mathbf{U} and \mathbf{V} by solving Equation (14) without binary constraints, then we initialize $\mathbf{B} \leftarrow \text{sign}(\mathbf{U})$, $\mathbf{D} \leftarrow \text{sign}(\mathbf{V})$, $\Phi \leftarrow \text{Equation (22)}$, $\mathcal{H} \leftarrow \text{Equation (24)}$;

2 **repeat**

3 **for** $i \in \{1, \dots, n\}$ **do**

4 **for** $j \in I_u^+$ and $k \in I_u^-$ **do**

5 Compute φ_{uij} by Equation (22) ;

6 Update \mathbf{b}_u by sloving the objective (17) by Xpress solver;

7 **for** $i \in \{1, \dots, m\}$ **do**

8 **for** $u \in \mathcal{U}_i^+$ and $k \in I_u^-$ **do**

9 Compute η_{uji} by Equation (24) ;

10 Update \mathbf{d}_i by sloving the objective (19) by Xpress solver;

11 **until** \mathcal{L}_{dsml} is convergent;

4 EXPERIMENTS

This section puts forward a new metric learning method SIML (continuous version of DSIML). In this section, we evaluate the performance of the hashing-based recommendation framework DSIML by comparing it with the competitive hashing-based recommendations DCF, DPR, CCCF, LightRec, and the latest PHD [2]. Besides, we evaluate the effectiveness of the proposed metric learning method SIML(continuous version of DSIML) by comparing with state-of-the-are metric learning recommendation methods CML [5], SML [10], one representative pairwise ranking-based recommendation framework BPR [17], the competitive baseline LightGCN [3], and the latest ranking-based recommendation method CPR [25].

we aim to answer the following questions to evaluate the effectiveness of the proposed method:

- **RQ1:** Does the proposed DSIML improve the recommendation performance compared with hashing-based recommender systems?
- **RQ2:** Does the proposed metric learning with scale-invariant margin improve the recommendation performance compared with state-of-the-art metric learning methods for recommendations?
- **RQ3:** Does the proposed DSIML speed up the online recommendation compared with one continuous representative method BPR?
- **RQ4:** Does the scale-invariant margin effectively cope with the imbalanced data problem?
- **RQ5:** How do the hyper-parameters λ influence the recommendation performance?

4.1 Experimental Settings

4.1.1 *Datasets.* we evaluate the performance of the proposed method and baselines with three real-world datasets: CDs, Movies, and Books, which are subset datasets of Amazon². Amazon datasets cover user interactions on items and item content on 24 product categories. For datasets we used in this section, we remove users and items with less than 20 ratings. Then, we convert explicit ratings into implicit feedback, where ‘1’ represents a user rated an item that is considered

²<http://jmcauley.ucsd.edu/data/amazon/>

as a positive item. Items with a value ‘0’ are considered as negative items. The statistics of datasets are listed in Table 1.

Table 1. Statistics of datasets.

Dataset	Users	Items	Ratings	Sparsity
CDs	25,400	24,904	43,903	99.99%
Movies	18,128	11,252	784,382	99.62%
Books	603,374	348,957	8,575,000	99.99%

we randomly split positive implicit feedback into 80% for training and the remaining 20% for testing. For each positive item, we randomly sample 5 negative items for training inspired by NCF [4]. 80% positive ratings together with these sampled negative ratings make up the training dataset. we report the performance of top- k recommendations under two metrics: *normalized discounted cumulative gain (NDCG@k)* [9] and *hit ratio (HR@k)* [31] since they have been widely used for evaluating the performance of ranking tasks.

4.1.2 Hyper-parameter Settings. we tune the optimal hyper-parameters for the proposed method by the grid search. Specifically, we search the hyper-parameter λ for each dataset within $\{1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^0, 1e^1, 1e^2\}$. For the scale-invariant margin $\gamma = \tan \beta$, due to the limitation of the smallest angle $\alpha \leq 60^\circ$ in the triangle $\Delta o'i'j$, and $\alpha \leq \beta$, so we tune the margin γ from $\{0.2, 0.5, \dots, 1.7\}$ due to $\tan(60^\circ) \approx 1.7$. we report the results of tuning the two hyper-parameters in the following experiments on Movies

4.1.3 Baselines. As introduced above, to evaluate the accuracy improvement of the proposed discrete method DSIML, we compare the performance of DSIML with the state-of-the-art hashing-based methods DCF, CCCF, LightRec, and PHD. Besides, to evaluate the effectiveness of metric learning proposed in this chapter, we compare the performance of the continuous version of SIML with the state-of-the-art five continuous frameworks BPR, CML, SML, LightGCN, and CPR.

- **DCF:** Discrete collaborative filtering [32] is the state-of-the-art hashing-based recommender system, which directly learns hash codes by optimizing a discrete optimization problem.
- **CCCF:** Compositional coding for collaborative filtering [13] constructs a compositional hashing method with G components for each user and item and optimizes a discrete problem directly.
- **LightRec:** LightRec [12] is a lightweight recommender system which is constituted by a backbone of B codebooks, each of which is composed of W latent codewords.
- **PHD:** Hansen et.al puts forward a Variational Hashing (VH) model [2] that mainly composed of a new measurement of dissimilarity, Projected Hamming Dissimilarity (PHD) between items in Hamming subspace. These dissimilarities are considered as the important weights of bits in hash codes.
- **BPR:** Bayesian Personalized Ranking framework [17] is a ranking-based recommendation based on Matrix Factorization, which directly optimizes ranking based evaluation metrics with Bayesian.
- **CML** Collaborative metric learning [5] learns a joint metric space to encode user-user relations together with relations of user and positive item. It is the first work that combines metric learning into recommendations.
- **SML** Symmetric metric learning [10] aims to search different optimal margin for each user and each item with a symmetric hinge loss.

- **LightGCN** LightGCN [3] constructs Graph Convolution Network (GCN) based on user-item interactive data, and exploits both user-item interactions and user-user relations by a second-order embedding smoothness step.
- **CPR** Cross Pairwise Ranking (CPR) proposes an unbiased recommendation by changing point-wise and pairwise rating loss.

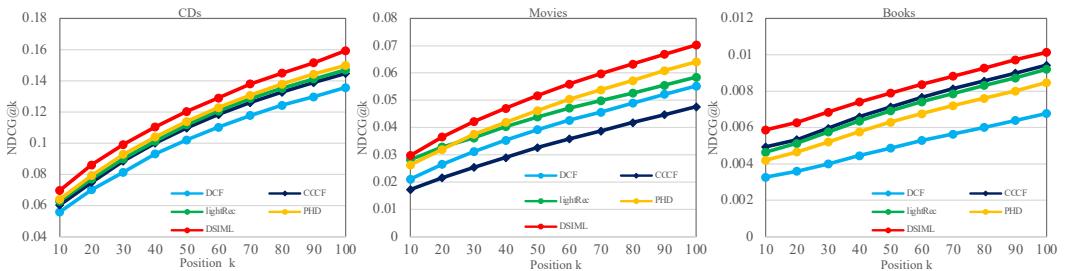


Fig. 3. The performance of the proposed discrete DSIML compared with state-of-the-art discrete recommendation methods on CDs, Movies, and Books under NDCG@k.

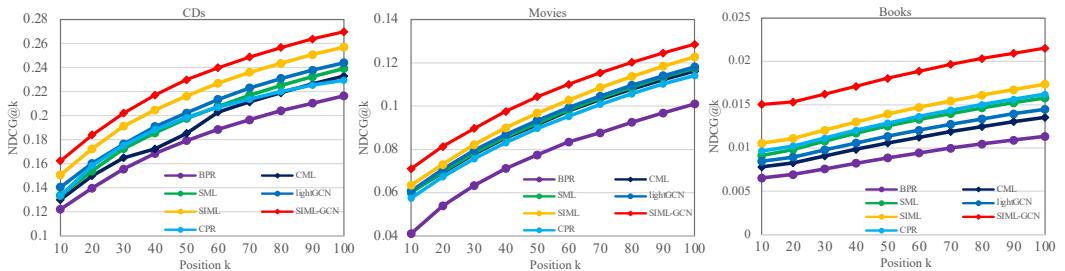


Fig. 4. The performance of the proposed continuous SIML and SIML-GCN (the variant on SIML) compared with state-of-the-art metric learning based recommendation models and other competitive continuous recommendation methods on CDs, Movies and Books under NDCG@k.

4.2 Experimental Results and Analysis

In this section, we evaluate the performance of DSIML by comparing it with competitive hashing-based baselines and metric learning methods by answering five questions proposed at the beginning of Section 4.

4.2.1 The Performance Compared with Hashing-based Methods(RQ1). As shown in Figure 3 and Table 2, the performances under two metrics, NDCG@k and HR@k, of the proposed DSIML are superior to other hashing-based baselines, which indicates that the proposed scale-invariant metric learning is effective to capture users' preferences over items from unbalanced classes. Besides, it also evaluates that the combination of a pairwise ranking loss and scale-invariant metric learning loss effectively provides accurate recommendations. we formulate DSIML with the ranking loss instead of a rating loss used in DCF. That's why DSIML performs better than DCF. In addition, LightRec, CCCF, and PHD are formulated with rating loss objectives other than the ranking loss objective used in DSIML. That's why they perform inferior to DSIML.

Table 2. The performance of DSIML compared with state-of-the-art discrete methods on CDs, Movies and Books under $HR@k$. The best performance is highlighted with bold text.

Dataset	CDs			Movies		
Metric	$HR@10$	$HR@50$	$HR@100$	$HR@10$	$HR@50$	$HR@100$
DCF	0.0504	0.1768	0.2821	0.0175	0.0680	0.1184
CCCF	0.0543	0.0191	0.0301	0.0142	0.0567	0.1043
LightRec	0.0562	0.1936	0.3038	0.0226	0.0705	0.1160
PHD	0.0578	0.1965	0.3095	0.0213	0.0774	0.1332
DSIML	0.0621	0.2038	0.3207	0.0236	0.0859	0.1439

Dataset	Books					
Metric	$HR@10$	$HR@30$	$HR@50$	$HR@70$	$HR@90$	$HR@100$
DCF	0.0019	0.0045	0.0068	0.0088	0.0109	0.0119
CCCF	0.0028	0.0066	0.0098	0.0124	0.0147	0.0159
LightRec	0.0025	0.0063	0.0093	0.0118	0.0137	0.0145
PHD	0.0024	0.0059	0.0087	0.0112	0.0134	0.0148
DSIML	0.0034	0.0074	0.0102	0.0127	0.0152	0.0163

Table 3. The performance of continuous SIML and SIML-GCN (the variant on SIML) compared with state-of-the-art metric learning based recommendation models and other competitive continuous recommendation methods on CDs, Movies and Books under $HR@k$. The best performance is highlighted with bold text.

Dataset	CDs			Movies		
Metric	$HR@10$	$HR@50$	$HR@100$	$HR@10$	$HR@50$	$HR@100$
BPR	0.1013	0.2768	0.4042	0.0368	0.1187	0.1854
CML	0.1113	0.2828	0.4243	0.0470	0.1427	0.2223
SML	0.1145	0.2970	0.4283	0.0461	0.1405	0.2183
LightGCN	0.1186	0.3047	0.4357	0.0476	0.1452	0.2260
CPR	0.1157	0.2992	0.4314	0.0468	0.1440	0.2216
SIML	0.1281	0.3220	0.4501	0.0505	0.1520	0.2346
SIML-GCN	0.1352	0.3380	0.4643	0.0547	0.1628	0.2477

Dataset	Books					
Metric	$HR@10$	$HR@30$	$HR@50$	$HR@70$	$HR@90$	$HR@100$
BPR	0.0036	0.0082	0.0117	0.0148	0.0174	0.0186
CML	0.0044	0.0098	0.0140	0.0176	0.0208	0.0221
SML	0.0052	0.0115	0.0162	0.0202	0.0237	0.0252
LightGCN	0.0048	0.0105	0.0149	0.0187	0.0220	0.0235
CPR	0.0054	0.0118	0.0167	0.0209	0.0243	0.0260
SIML	0.0049	0.0118	0.0171	0.0212	0.0249	0.0266
SIML-GCN	0.0141	0.0191	0.0232	0.0268	0.0298	0.0312

4.2.2 *The Performance Compared with Metric Learning Methods(RQ2)*. As shown in Figure 4 and Table 3, they reflect the performance of the proposed SIML and its invariant SIML-GCN under two metrics $NDCG@k$ and $HR@k$ are significantly superior to two other competing metric learning based recommendation models, CML and SML. For SIML-GCN, we construct a Graph Convolution

Table 4. The Effectiveness of Speeding up the Online Recommendation

Datasets	BPR(Seconds)	DSIML(Seconds)	Speedup ratio
Yelp	2721.84	213.18	12.76 times ↑
Books	9082.41	512.26	17.73 times ↑
Movies	41.65	5.31	7.84 times ↑
MovieLens	16.83	3.77	4.46 times ↑

Network (GCN) based on ratings and impose the scale-invariant margin loss on embeddings of users and items.

For two state-of-the-art metric-learning-based recommendations, CML and SML, SML achieves better performances than CML in most cases due to two drawbacks that existed in CML. But the phenomena is not consistent when evaluated on these three datasets. The performance of SML on Movies is significantly better than on CDs and Books because CDs and Books are more sparse than Movies. SML learns adaptive margins for each user and item, which easily leads to overfitting when the data is sparse.

Besides, Figure 4 and Table 3 also show that the performances of SML are evidently superior to the pairwise ranking-based recommendation BPR, CPR. Because BPR and CPR are MF-based methods that violate the triangle inequality according to [21], which leads to the failure of capturing more preference information. Therefore, metric learning with a scale-invariant margin provides accurate top- k personalized recommendations more effectively than pairwise ranking-based models.

By comparing with LightGCN, LightGCN can not only capture the first-order user-user relation, and user-item relation but also capture the second-order relations. That's why LightGCN performs better than the state-of-the-art metric learning methods and pairwise ranking-based methods. LightGCN can achieve comparable performance to SML. By formulating the proposed metric learning with GCN, i.e., SML-GCN, the performance of the proposed method is much better than LightGCN.

Efficiency Comparison with Continuous Methods (RQ3). As discussed in hashing-based baselines [13, 32, 35], DSIML maps users and items in d -dimension Hamming space, and so the online recommendation is significantly accelerated with the fast similarity search by XOR bit operations [32]. Table 4 shows the efficiency performance of the online recommendation on three datasets by comparing it with the continuous method BPR. Due to continuous methods obtaining real-valued representations, continuous methods perform similarly for the online recommendation stage, and thus we choose the representative BPR for comparison. Table 4 shows the significant superiority of the hashing-based method DSIML for providing efficient online recommendations.

The Effectiveness of Scale-invariant Margin for Imbalanced Data (RQ4). Figure 5 shows the effectiveness of the proposed scale-invariant margin for items from imbalanced classes. we choose four categories of movies on Movies for evaluation, as shown in the right figure. Experimental results on these imbalanced items validate that the proposed scale-invariant margin is more effective in providing accurate recommendations than the fixed margin used in CML and SML.

Figure 6 shows how does the scale-invariant margin $\gamma = \tan \beta$ affect the recommendation performance of SML. we conclude that greater margin γ leads to inferior performance under both two metrics, which is reasonable because greater margin leads to greater angle α at the negative sample point in Figure 2, so the recommendation model do not capture fine grained preference difference between negative items and positive items, which leads to poor performance in recommendations.

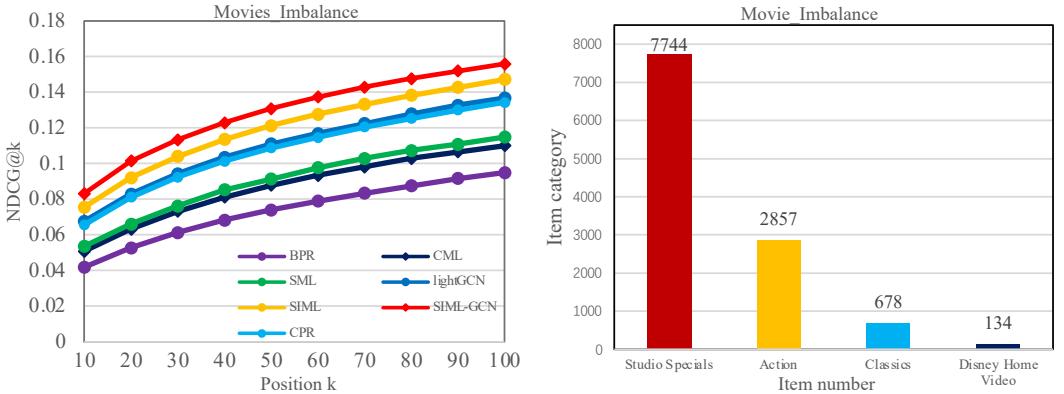


Fig. 5. The performance of the proposed SIML on imbalanced data selected on Movies under NDCG@k.

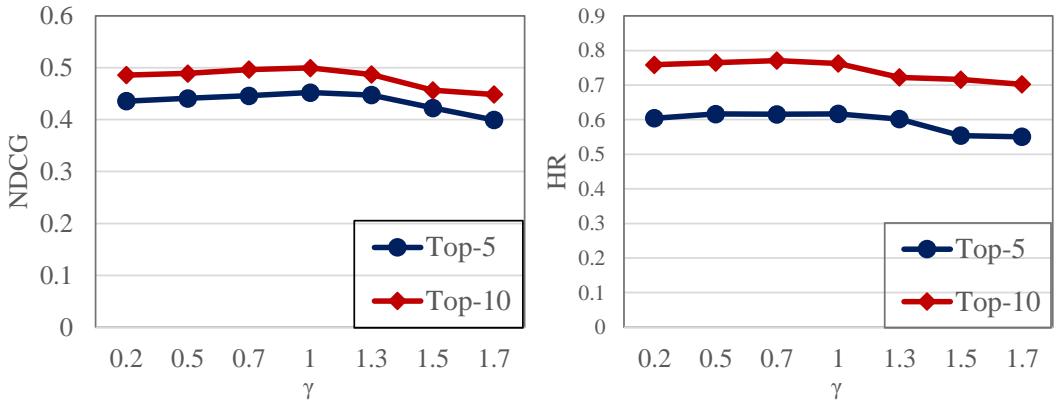


Fig. 6. The effect of scale-invariant margin γ of SIML for recommendation performances on Movies.

A smaller margin leads to better performance because it enforces a stronger margin to the angle α . However, the performance is stable when the γ increases from 1.3 to 1.7 because a stronger margin makes the optimization problem intractable. Thus the model can only find a locally optimal solution. So we set $\gamma = 1$ on the Movies dataset in the experiments.

The Influence of Hyper-parameter λ (RQ5). Figure 7 displays the result of tuning the hyper-parameter λ within the range of $\{e^{-3}, \dots, e^2\}$ by the grid search. we conclude that the performance of recommendations is relatively stable. To achieve better top- k recommendation performance, we choose $\lambda = 1$ on the Movies dataset because it achieves the best performance.

5 CONCLUSION

In this paper, we propose a discrete scale-invariant metric learning (DSIML) for recommendations. Specifically, we first present a scale-invariant margin on angles at negative points and derive a scale-invariant triple hinge loss based on the angle margin. By optimizing the objective, negative items are dragged away from both users and positive items. Besides, the angle margin is invariant with different unbalanced classes. Finally, we formulate the proposed model by incorporating the scale-invariant loss with a pairwise ranking loss and imposing binary constraints on users and items to learn hash codes in Hamming subspace. Experiments on DSIML, its continuous version SIML,

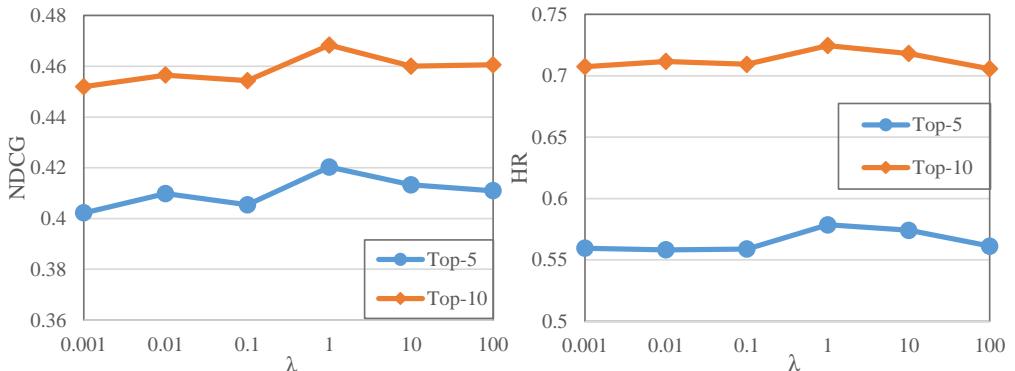


Fig. 7. The influence of hyper parameter λ of SIML for recommendation performances on Movies.

and its variation SIML-GCN show that the proposed scale-invariant metric learning respectively outperforms hashing-based baselines, metric learning methods, and the competitive LightGCN.

REFERENCES

- [1] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. 2013. One-pass AUC optimization. In *International Conference on Machine Learning*. 906–914.
- [2] Christian Hansen, Casper Hansen, Jakob Grue Simonsen, and Christina Lioma. 2021. Projected hamming dissimilarity for bit-level importance coding in collaborative filtering. In *Proceedings of the Web Conference 2021*. 261–269.
- [3] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.
- [4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.
- [5] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th international conference on world wide web*. 193–201.
- [6] Lin Hu, Junxiang Peng, Yan Zhang, Hong Wen, Shuai Tan, and Jiabing Fan. 2022. Artificial noise assisted interference alignment for physical layer security enhancement. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 4148–4153.
- [7] Shanshan Huang, Shuaiqiang Wang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. 2015. Listwise collaborative filtering. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 343–352.
- [8] Tommi S Jaakkola and Michael I Jordan. 1997. A variational approach to Bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*. PMLR, 283–294.
- [9] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 243–250.
- [10] Mingming Li, Shuai Zhang, Fuging Zhu, Wanhai Qian, Liangjun Zang, Jizhong Han, and Songlin Hu. 2020. Symmetric metric learning with adaptive margin for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 4634–4641.
- [11] Roger Zhe Li, Julián Urbano, and Alan Hanjalic. 2021. New Insights into Metric Optimization for Ranking-based Recommendation. (2021), 145–154.
- [12] Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. LightRec: A Memory and Search-Efficient Recommender System. In *Proceedings of The Web Conference 2020*. 695–705.
- [13] Chenghao Liu, Tao Lu, Xin Wang, Zhiyong Cheng, Jianling Sun, and Steven CH Hoi. 2019. Compositional coding for collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 145–154.
- [14] Chen Ma, Liheng Ma, Yingxue Zhang, Ruiming Tang, Xue Liu, and Mark Coates. 2020. Probabilistic metric learning with adaptive margin for top-K Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1036–1044.
- [15] Alistair Moffat, Falk Scholer, and Paul Thomas. 2012. Models and metrics: IR evaluation as a user process. In *proceedings of the seventeenth Australasian document computing symposium*. 47–54.

- [16] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 273–282.
- [17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI'09*. AUAI Press, 452–461.
- [18] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *CVPR*. 37–45.
- [19] 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 sixth ACM conference on Recommender systems*. ACM, 139–146.
- [20] Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*. 269–272.
- [21] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). *arXiv preprint arXiv:1405.5869* (2014).
- [22] Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*. 1857–1865.
- [23] Nathan Srebro, Jason DM Rennie, and Tommi S Jaakkola. 2004. Maximum-Margin Matrix Factorization.. In *NIPS*, Vol. 17. Citeseer, 1329–1336.
- [24] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2289–2298.
- [25] Qi Wan, Xiangnan He, Xiang Wang, Jiancan Wu, Wei Guo, and Ruiming Tang. 2022. Cross Pairwise Ranking for Unbiased Item Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2370–2378.
- [26] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. 2017. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*. 2593–2601.
- [27] Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. 2008. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems*. 1593–1600.
- [28] Bin Wu, Tianren Shi, Lihong Zhong, Yan Zhang, and Yangdong Ye. 2023. Graph-coupled time interval network for sequential recommendation. *Information Sciences* 648 (2023), 119510.
- [29] Liwei Wu, Cho-Jui Hsieh, and James Sharpnack. 2018. Sql-rank: A listwise approach to collaborative ranking. In *International Conference on Machine Learning*. PMLR, 5315–5324.
- [30] Hui Xu, Changyu Li, Yan Zhang, Lixin Duan, Ivor W Tsang, and Jie Shao. 2022. Metacar: Cross-domain meta-augmentation for content-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2022), 8199–8212.
- [31] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. 2013. Lcars: a location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 221–229.
- [32] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *Proceedings of SIGIR'16*, Vol. 16.
- [33] Juling Zhang, Guowu Yang, William NN Hung, Yan Zhang, and Jinzhao Wu. 2019. An efficient NPN Boolean matching algorithm based on structural signature and Shannon expansion. *Cluster Computing* 22 (2019), 7491–7506.
- [34] Yan Zhang, Changyu Li, Ivor W Tsang, Hui Xu, Lixin Duan, Hongzhi Yin, Wen Li, and Jie Shao. 2022. Diverse preference augmentation with multiple domains for cold-start recommendations. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2942–2955.
- [35] Yan Zhang, Defu Lian, and Guowu Yang. 2017. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- [36] Yan Zhang, Haoyu Wang, Defu Lian, Ivor W Tsang, Hongzhi Yin, and Guowu Yang. 2018. Discrete ranking-based matrix factorization with self-paced learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2758–2767.