Seongmin Park Sungkyunkwan University Suwon, Republic of Korea psm1206@skku.edu Mincheol Yoon Sungkyunkwan University Suwon, Republic of Korea yoon56@skku.edu

Hogun Park Sungkyunkwan University Suwon, Republic of Korea hogunpark@skku.edu Jae-woong Lee Sungkyunkwan University Suwon, Republic of Korea jwlee.icc@skku.edu

Jongwuk Lee\* Sungkyunkwan University Suwon, Republic of Korea jongwuklee@skku.edu

# ABSTRACT

Collaborative filtering (CF) is a pivotal technique in modern recommender systems. The learning process of CF models typically consists of three components: interaction encoder, loss function, and negative sampling. Although many existing studies have proposed various CF models to design sophisticated interaction encoders, recent work shows that simply reformulating the loss functions can achieve significant performance gains. This paper delves into analyzing the relationship among existing loss functions. Our mathematical analysis reveals that the previous loss functions can be interpreted as alignment and uniformity functions: (i) the alignment matches user and item representations, and (ii) the uniformity disperses user and item distributions. Inspired by this analysis, we propose a novel loss function that improves the design of alignment and uniformity considering the unique patterns of datasets called Margin-aware Alignment and Weighted Uniformity (MAWU). The key novelty of MAWU is two-fold: (i) margin-aware alignment (MA) mitigates user/item-specific popularity biases, and (ii) weighted uniformity (WU) adjusts the significance between user and item uniformities to reflect the inherent characteristics of datasets. Extensive experimental results show that MF and LightGCN equipped with MAWU are comparable or superior to state-of-the-art CF models with various loss functions on three public datasets.

## CCS CONCEPTS

• Information systems  $\rightarrow$  Recommender systems; Collaborative filtering.

## **KEYWORDS**

Collaborative filtering, loss function, theoretical analysis, alignment and uniformity

\*Corresponding author

CIKM '23, October 21-25, 2023, Birmingham, United Kingdom

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0124-5/23/10...\$15.00 https://doi.org/10.1145/3583780.3615086

#### **ACM Reference Format:**

Seongmin Park, Mincheol Yoon, Jae-woong Lee, Hogun Park, and Jongwuk Lee. 2023. Toward a Better Understanding of Loss Functions for Collaborative Filtering. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25,* 2023, Birmingham, United Kingdom. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3583780.3615086

## **1 INTRODUCTION**

Recommender systems [30] are widespread in various Web applications (e.g., YouTube, Amazon, Netflix, and Spotify) to assist users in navigating the abundance of information available during the decision-making process. Collaborative filtering (CF) [13, 14, 17, 19, 23, 27, 29, 32, 39, 47, 49, 61] is a core technique for identifying meaningful collaborative signals among users/items and predicting hidden user preferences on items. The primary advantage of CF is that it only uses past user-item interactions without the need for auxiliary information about users/items. Owing to its simplicity and efficacy, numerous CF models have been developed for various domains and tasks in recommender systems.

The learning process of CF models primarily consists of three key components: *interaction encoder*, *loss function*, and *negative sampling* [25]. Many studies [1, 2, 13, 17, 19, 23, 26, 31–34, 45, 53] have focused on designing effective interaction encoders to capture complex correlations across users and items. However, some works [5, 6, 9] have pointed out that the performance of these encoders may be overestimated under certain experimental conditions. Notably, recent studies [25, 37, 46, 56] have highlighted that reformulating loss functions is significant for improving the performance of CF models. In particular, MF [17] equipped with CCL [25], SSM [46], DirectAU [37], and BC [56] loss functions has yielded competitive performance compared to state-of-the-art CF models. However, there is no previous study that represents an in-depth analysis of existing loss functions mathematically.

Before diving into the analysis, we first categorize existing loss functions [11, 16, 17, 21, 25, 29, 37, 46, 56, 62] into four types, *i.e.*, *pointwise*, *pairwise*, *setwise*, and *alignment and uniformity (AU)* loss functions, as summarized in Table 1. The pointwise loss functions are divided into two loss terms according to positive and negative labels of user-item interactions. The pairwise and setwise loss functions employ a single loss term with a triplet (user, positive item, negative item(s)). While the pairwise loss function uses a single

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.

Туре	Name	Similarity or distance	Loss function
	BCE [17]	$s(u,i) = \tilde{f(u)}^{\mathrm{T}} \tilde{f(i)}$	$-\sum_{(u,i)\in\mathcal{D}}\log\sigma\left(s(u,i)\right)-\sum_{(u,j)\notin\mathcal{D}}\log\left(1-\sigma(s(u,j))\right)$
Pointwise	MCL [11]	$d(u,i) =   \tilde{f(u)} - \tilde{f(i)}  _2^2$	$\frac{1}{\alpha} \log \left( 1 + \frac{1}{m} \sum_{(u,i) \in \mathcal{D}} e^{\alpha \left( d(u,i) + \lambda_p \right)} \right) + $
			$\frac{1}{\beta} \log \left( 1 + \frac{1}{m} \sum_{(u,j) \notin \mathcal{D}} e^{-\beta(d(u,j) + \lambda_n)} \right)$
	UIB [62]	$s(u,i) = \tilde{f(u)}^{\mathrm{T}} \tilde{f(i)}$	$-\sum_{(u,i)\in\mathcal{D}}\log\sigma\left(s(u,i)-b_u\right)-\alpha\sum_{(u,j)\notin\mathcal{D}}\log\sigma\left(b_u-s(u,j)\right)$
	BPR [29]	$s(u, i) = \tilde{f(u)}^{\mathrm{T}} \tilde{f(i)}$	$-\sum_{(u,i,j)\in\mathcal{T}}\log\sigma\left(s(u,i)-s(u,j)\right)$
Pairwise	CML [16]	$d(u,i)=  \tilde{f(u)}-\tilde{f(i)}  _2^2$	$\sum_{(u,i,j)\in\mathcal{T}} \left[ d(u,i) - d(u,j) + M \right]_+$
	SML [21]	$d(u,i)=  \tilde{f(u)}-\tilde{f(i)}  _2^2$	$\sum_{(u,i,j)\in\mathcal{T}} \left( [d(u,i) - d(u,j) + M_u]_+ + [d(u,i) - d(i,j) + M_i]_+ \right) + \lambda \mathcal{L}_{AM}$
	CCL [25]	$s(u,i) = \tilde{f(u)}^{\mathrm{T}} \tilde{f(i)}$	$\sum_{(u,i,\mathcal{N}_u)\in\mathcal{T}_s} \left(1-s(u,i)+\frac{w}{ \mathcal{N}_u }\sum_{j\in\mathcal{N}_u}[s(u,j)-M]_+\right)$
Setwise	SSM [46]	$s(u,i) = \tilde{f(u)}^{\mathrm{T}} \tilde{f(i)}$	$-\sum_{(u,i,\mathcal{N}_u)\in\mathcal{T}_s}\log\frac{\exp(s(u,i)/\tau)}{\exp(s(u,i)/\tau)+\sum_{j\in\mathcal{N}_u}\exp(s(u,j)/\tau)}$
	BC [56]	$s(u,i) = \cos(\hat{\theta}_{ui})$	$-\sum_{(u,i,\mathcal{N}_u)\in\mathcal{T}_s}\log\frac{\exp(\cos(\hat{\theta}_{ui}+M_{ui})/\tau)}{\exp(\cos(\hat{\theta}_{ui}+M_{ui})/\tau)+\sum_{j\in\mathcal{N}_u}\exp(\cos(\hat{\theta}_{uj})/\tau)}$
Align. & Unif.	DAU [37]	$d(u, i) =   \tilde{f(u)} - \tilde{f(i)}  _2^2$	$\mathbb{E}_{(u,i)\sim p_{pos}}d(u,i)+\gamma\left(\log\mathbb{E}_{(u,u')\sim p_{user}}e^{-2\cdot d(u,u')}+\log\mathbb{E}_{(i,i')\sim p_{item}}e^{-2\cdot d(i,i')}\right)$

Table 1: Architecture comparison between CF losses categorized by four types. UIB [62] uses BPR [29] as its backbone. All the notations in loss functions follow the notations of each original paper. Due to limited space, DirectAU is denoted as DAU.

negative item, the setwise loss function utilizes a set of negative items in one triplet. Meanwhile, the AU loss function receives positive user-item pairs to compute the alignment function and utilizes all user and item samples in a batch to compute the uniformity function.

Then, we investigate the mathematical relationships among the existing loss functions. We mainly consider seven loss functions such as BPR [29], CML [16], SML [21], CCL [25], SSM [46], BC [56], and DirectAU [37] except for the pointwise loss functions<sup>1</sup>. Our analysis reveals the following key observations: (i) SSM [46] is regarded as the fundamental loss function among pairwise and setwise loss functions. Moreover, BC [56] represents an improvement over the loss function of SSM [46] by introducing the interactionspecific angular margins. (ii) SSM [46] is also interpreted as DirectAU [37], consisting of two loss functions, i.e., alignment and uniformity for contrastive representation learning. These observations highlight two main ways in which the performance of recent recommendation systems can be improved. First, BC [56] can learn angular margins to account for the popularity bias inherent in each user-item interaction. Second, DirectAU [37] does not require negative sampling strategies, which saves additional costs.

The above analysis has led us to design an improved loss function, called *Margin-aware Alignment and Weighted Uniformity (MAWU)*, leveraging the strengths of both BC [56] and DirectAU [37]. Specifically, it consists of the following two loss functions.

• Margin-aware alignment (MA): It enhances the existing alignment function by introducing user/item-specific margins. It can further mitigate the biases from the active users and popular items, requiring fewer parameters during the learning. As

a result, more robust predictions are expected, particularly on sparser interaction graphs.

• Weighted uniformity (WU): It employs different hyperparameters to adjust the significance of user and item uniformities. For instance, if the user distribution is more uniform than the item distribution, we emphasize user uniformity more. Despite its simplicity, it easily reflects the inherent statistics of datasets, *i.e.*, Gini indices of user and item, improving the performance of the original uniformity function.

The main advantage of MAWU is two-fold: (i) It can learn better user/item representations by directly optimizing alignment and uniformity. (ii) It adaptively reflects the unique characteristics of datasets by using the margin for alignment and the adjusted importance for uniformity.

The key contributions of this paper are summarized as follows:

- Through a mathematical analysis of seven loss functions, we derive that SSM is the fundamental loss function, which is also expressed in the form of optimizing alignment and uniformity. Moreover, we suggest the key potential of combining BC [56] and DirectAU [37]. (Section 3)
- We propose a Margin-aware Alignment and Weighted Uniformity (MAWU) loss function. (i) Margin-aware alignment (MA) efficiently learns with the angular margin for mitigating user/item popularity biases, and (ii) weighted uniformity (WU) is used to adjust the significance of the user/item uniformity. (Section 4)
- Extensive experimental results demonstrate that MAWU outperforms ten existing loss functions and eight state-of-the-art CF models on three benchmark datasets. (Sections 5-6)

<sup>&</sup>lt;sup>1</sup>The pointwise loss functions are difficult to integrate mathematically with other loss functions because they are split into two loss terms based on the labels.

## 2 BACKGROUND

## 2.1 Formulation of CF models

**Notations.** Let  $\mathcal{U}$  and  $\mathcal{I}$  denote a set of users and a set of items, respectively. Assuming the implicit setting, we define a user-item interaction matrix  $\mathbf{R} \in \{0, 1\}^{|\mathcal{U}| \times |\mathcal{I}|}$ . If  $r_{ui} = 1$ , it indicates user u has interacted with item i. Otherwise, *i.e.*,  $r_{ui} = 0$ , there is no interaction between user u and item i. Given user u, we define  $\mathcal{I}_u^+ = \{i \in \mathcal{I} \mid r_{ui} = 1\}$  as a set of items interacted by user u. Let  $\mathcal{D} = \{(u, i) \mid u \in \mathcal{U} \land i \in I_u^+\}$  denote a set of observed user-item pairs, and  $\mathcal{T} = \{(u, i, j) \mid u \in \mathcal{U} \land i \in \mathcal{I}_u^+ \land j \in \mathcal{I} \setminus \mathcal{I}_u^+\}$  denote a set of triplets for positive and negative user-item pairs. Here,  $\mathcal{T}$  is extended for a set of negative pairs. Let  $\mathcal{T}_s = \{(u, i, \mathcal{N}_u) \mid u \in \mathcal{U} \land i \in \mathcal{I}_u^+ \land \mathcal{N}_u \subset \mathcal{I} \setminus \mathcal{I}_u^+\}$  denote a set of triplets, where  $\mathcal{N}_u$  is a set of negative items of user u. The goal of CF models is to infer a prediction score  $\hat{r}_{ui}$  for unobserved user-item pair (u, i) and recommend the items with the highest scores for user u.

Three key components of CF models. As discussed in existing work [25], the learning process of CF models consists of three parts, *i.e., interaction encoder, loss function,* and *negative sampling.* (i) Interaction encoder: It learns user/item embeddings to capture collaborative signals. (ii) Loss function: CF models commonly utilize four types of loss functions such as *pointwise, pairwise, setwise,* and *alignment and uniformity (AU)* loss functions. Each loss function will be described in detail. (iii) Negative sampling: Because most entries in **R** are unobserved, it is common to perform negative sampling for unobserved user-item pairs to improve training efficiency.

Although designing the interaction encoder is vital to capture complex and various correlations across users/items, recent studies [8, 11, 25, 37, 46, 56] report that loss functions and negative sampling are also beneficial to improve overall performance without designing complicated models. Notably, some studies [25, 37, 46, 56] show that the simple MF model beats complicated neural models by simply modifying loss functions. Despite the recent progress, there is no formal study that investigates the relationship between various loss functions. This paper thus focuses on analyzing the relationship between various loss functions and proposes a novel loss function for CF models.

#### 2.2 Four Categories of Designing Loss Functions

Table 1 depicts various loss functions for CF models. Each loss function expresses the relevance score between user *u* and item *i* as the cosine similarity or Euclidean distance. We denote the cosine similarity as  $s(u, i) = f(\tilde{u})^{T} f(\tilde{i})$  and Euclidean distance as  $d(u, i) = ||f(\tilde{u}) - f(\tilde{i})||_{2}^{2}$ , where f(u) and  $f(i) \in \mathbb{R}^{d}$  are user and item embedding vectors, respectively, and  $f(\tilde{\cdot})$  means normalized representations. Existing loss functions are categorized as follows.

• **Pointwise loss function**: Because user-item pairs have either positive or negative values, it is natural to optimize them as the label of user-item pairs, *e.g.*, BCE [17], MCL [11], and UIB [62].

$$\mathcal{L}_{Pointwise} = \sum_{(u,i)\in\mathcal{D}} \delta_{ui} + \sum_{(u,j)\notin\mathcal{D}} \delta_{uj}, \tag{1}$$

where  $\delta_{ui}$  and  $\delta_{uj}$  denote a loss for positive pair (u, i) and negative pair (u, j), respectively. For BCE [17],  $\delta_{ui} = -\log \sigma(s(u, i))$  and  $\delta_{uj} = -\log(1 - \sigma(s(u, j)))$ .

• Pairwise loss function: Because the pairwise loss function is designed to derive the maximum posterior estimator for personalized ranking, it is more suitable for addressing the top-*N* recommendation problem. It includes BPR [29], CML [16], and SML [21]. Because the number of triple pairs is proportional to the quadratic complexity for the number of items, it requires slow training time to converge.

$$\mathcal{L}_{Pairwise} = \sum_{(u,i,j)\in\mathcal{T}} \delta_{uij},\tag{2}$$

where  $\delta_{uij}$  means a loss for a triplet (u, i, j). For BPR [29],  $\delta_{uij} = -\log \sigma(s(u, i) - s(u, j))$ .

• Setwise loss function: As the generalization of the pairwise loss function, it considers multiple negative item pairs at once. It includes CCL [25], SSM [46], and BC [56]. Since we typically sample *n* negative items for each positive pair, it is more effective to consider a positive item and a set of negative items with equal weight and put them in one triplet.

$$\mathcal{L}_{Set\,wise} = \sum_{(u,i,\mathcal{N}_u)\in\mathcal{T}_s} \delta_{ui\,\mathcal{N}_u}.$$
(3)

If a single negative item is used (*i.e.*,  $|\mathcal{N}_u| = 1$ ), it is equivalent to the pairwise loss function. For CCL [25],  $\delta_{ui\mathcal{N}_u} = (1 - s(u, i)) + \frac{w}{|\mathcal{N}_u|} \sum_{j \in \mathcal{N}_u} [s(u, j) - m]_+$ , where  $[\cdot]_+ = \max(0, \cdot)$ .

• Alignment and uniformity loss function: Recent studies [10, 37, 40] utilize the decomposition of the InfoNCE [36] loss function into two losses, such as alignment and uniformity (AU) loss.

$$\mathcal{L}_{AU} = \mathcal{L}_{Align} + \gamma \mathcal{L}_{Unif},\tag{4}$$

where  $\gamma$  denotes a hyperparameter to adjust alignment and uniformity. The alignment loss minimizes the distance between user *u* and item *i* for a positive pair (*u*, *i*). This accomplishes the goal of CF, which is to recommend similar items to users with similar preferences.

$$\mathcal{L}_{Align} = \mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} d(u,i), \tag{5}$$

where  $p_{pos}$  is the distribution for all positive user-item pairs. The uniformity loss makes all embedding vectors spread as far as possible on the hypersphere. This allows the hypersphere space to be maximally utilized to accommodate a variety of information. We compute the uniformity loss function on latent user and item spaces, respectively [37].

$$\mathcal{L}_{Unif} = \log \mathop{\mathbb{E}}_{(u,u')\sim p_{user}} e^{-2 \cdot d(u,u')} + \log \mathop{\mathbb{E}}_{(i,i')\sim p_{item}} e^{-2 \cdot d(i,i')},$$
(6)

where  $p_{user}$  and  $p_{item}$  represent the distribution for users and items, respectively. Besides, u' and i' are the in-batch user/item samples, respectively.

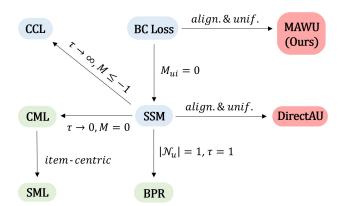


Figure 1: Relationship diagram of CF loss functions except for the pointwise loss function. The red, blue, and green backgrounds indicate alignment and uniformity, setwise, and pairwise loss functions, respectively.

## **3 MATHEMATICAL ANALYSIS**

In this section, we investigate the relationships between various CF loss functions, with a fundamental setwise loss function, SSM [46], as the reference. Figure 1 depicts the mathematical relationship across loss functions. Through our analysis, we reveal the following intriguing observations.

- Relationships between SSM [46] and pairwise loss functions: Because setwise loss functions regard multiple negative samples at once, we derive three pairwise loss functions, *i.e.*, CML [16], SML [21], and BPR [29], from SSM [46].
- Relationships between SSM [46] and other setwise loss functions: SSM [46] is induced to two setwise loss functions, *i.e.*, CCL [25] and BC [56]. Besides, BC improves SSM using the margin for popularity bias.
- Relationship between SSM [46] and DirectAU [37]: SSM [46] represents a contrastive loss, *i.e.*, InfoNCE. Thus, DirectAU [37] using alignment and uniformity is derived from SSM. Because DirectAU does not require negative sampling, it learns better user/item representations for CF models than SSM.

Through our analysis, we found two potential improvements: (i) BC [56] introduces a margin to adaptively optimize the useritem angular distance, and (ii) DirectAU [37] does not need any negative sampling strategies. However, there is no existing study that effectively combines the benefits of both loss functions.

#### 3.1 Analysis of Pairwise Loss Functions

We first analyze the relationship between SSM [46] and CML [16]. To derive CML [16] from SSM [46], we assume  $\tau \rightarrow 0^+$ , following the assumption of controlling  $\tau$  in the InfoNCE loss in [38]. Here,  $\tau$ is a temperature to control the skewness of the softmax distribution. When  $\tau \rightarrow 0^+$ , the SSM [46] is approximated to CML [16] using the hardest negative samples. For simplicity, we focus on deriving Seongmin Park, Mincheol Yoon, Jae-woong Lee, Hogun Park, & Jongwuk Lee

a single triplet  $\mathcal{T}_s(u, i, \mathcal{N}_u)$  in Eq. (19).

=

$$\lim_{\tau \to 0^{+}} -\log \frac{e^{s(u,i)/\tau}}{e^{s(u,i)/\tau} + \sum_{j \in \mathcal{N}_{u}} e^{s(u,j)/\tau}}$$
  
$$= \lim_{\tau \to 0^{+}} +\log \left( 1 + \sum_{j \in \mathcal{N}_{u}} e^{(s(u,j) - s(u,i))/\tau} \right)$$
(7)

$$\approx \lim_{\tau \to 0^+} + \log \left( 1 + \sum_{s(u,j) \ge s(u,i)}^{j} e^{(s(u,j) - s(u,i))/\tau} \right)$$
(8)

$$\approx \lim_{\tau \to 0^+} \frac{1}{\tau} \left[ s(u, j_{\max}) - s(u, i) \right]_+,$$
(9)

where  $j_{\text{max}}$  is the negative item with the highest similarity with user *u*. As  $\tau$  approaches zero in Eq. (7), the gap between a positive pair and a negative pair gets larger. Thus, Eq. (7) is approximated by Eq. (8) regarding the negative pairs satisfying  $s(u, j) \ge s(u, i)$ . Finally, Eq. (8) is expressed by Eq. (9), which considers only one pair with the hardest negative item  $j_{\text{max}}$ .

Furthermore, CML [16] has a margin. Assuming that the margin is zero, we finally derive the relationship between SSM and CML.

$$\lim_{\tau \to 0^+} \mathcal{L}_{SSM} \approx \mathcal{L}_{CML}(m=0) \text{ with the hardest negative item}$$

\_

$$\sum_{(u,i,j_{\max})\in\mathcal{T}} [s(u,j_{\max}) - s(u,i)]_+$$
(10)

$$\propto \sum_{(u,i,j_{\max})\in\mathcal{T}} \left[ d(u,i) - d(u,j_{\max}) \right]_+, \tag{11}$$

where  $s(u, i) = 1 - \frac{1}{2}(d(u, i))^2$  such that  $||d(u, i)||_F = 1$ . The similarity in Eq. (10) is thus converted to the Euclidean distance in Eq. (11). Although SML [21] is different from CML [16], it is interpreted as combining both user- and item-centric loss functions. By deriving SSM [46] in both the user and item directions, it is possible to analyze the relationship between SSM [46] and SML [21].

Lastly, we discuss how to derive BPR [29] from SSM [46]. When  $|\mathcal{N}_u| = 1$  and  $\tau = 1$ , it is equivalent to BPR [29]. Here,  $\tau = 1$  indicates that BPR does not employ hard negative filtering [38].

 $\mathcal{L}_{SSM}(|\mathcal{N}_{u}| = 1 \text{ and } \tau = 1) = \mathcal{L}_{BPR}$   $= -\sum_{(u,i,j)\in\mathcal{T}} \log \frac{e^{s(u,i)}}{e^{s(u,i)} + e^{s(u,j)}} = -\sum_{(u,i,j)\in\mathcal{T}} \log \sigma \left( s(u,i) - s(u,j) \right).$ (12)

#### 3.2 Analysis of Setwise Loss Functions

We analyze the relationship between SSM [46] and CCL [25]. We employ the assumption of  $\tau \rightarrow +\infty$  in [38]. When  $\tau \rightarrow +\infty$ , SSM is

interpreted as CCL without hard negative filtering.

$$\lim_{\tau \to +\infty} -\log \frac{e^{s(u,i)/\tau}}{e^{s(u,i)/\tau} + \sum_{j \in \mathcal{N}_u} e^{s(u,j)/\tau}}$$
$$= \lim_{\tau \to +\infty} -\frac{1}{\tau} s(u,i) + \log \sum_{j \in \{\mathcal{N}_u,i\}} e^{s(u,j)/\tau}$$
(13)

$$= \lim_{\tau \to +\infty} -\frac{1}{\tau} s(u, i) + \log \left( \frac{1}{|\mathcal{N}_{u}|} \sum_{j \in \{\mathcal{N}_{u}, i\}} e^{s(u, j)/\tau} - 1 + 1 \right) + \log |\mathcal{N}_{u}|$$
(14)

$$\approx \lim_{\tau \to +\infty} -\frac{1}{\tau} s(u, i) + \frac{1}{|\mathcal{N}_u|} \sum_{j \in \{\mathcal{N}_u, i\}} e^{s(u, j)/\tau} - 1 + \log |\mathcal{N}_u|$$
(15)

$$\approx \lim_{\tau \to +\infty} -\frac{|\mathcal{N}_u| - 1}{|\mathcal{N}_u|\tau} s(u, i) + \frac{1}{|\mathcal{N}_u|\tau} \sum_{j \in \mathcal{N}_u} s(u, j) + \log|\mathcal{N}_u| + \frac{1}{|\mathcal{N}_u|}$$
(16)

$$\approx \lim_{\tau \to +\infty} -\frac{1}{\tau} s(u, i) + \frac{1}{|\mathcal{N}_u|\tau} \sum_{j \in \mathcal{N}_u} s(u, j) + \log |\mathcal{N}_u|.$$
(17)

Here, Eq. (16) is approximately derived by utilizing Taylor expansion of log(1 + x) and exp(x) in Eq. (14) and Eq. (15), respectively, *i.e.*, log(1+x)  $\approx$  x and exp(x)  $\approx$  1+x, for  $|x| \ll 1$ . As  $\tau$  approaches infinity,  $(\frac{1}{|N_u|} \sum_{j \in \{N_u, i\}} e^{s(u,j)/\tau} - 1)$  in Eq. (14) and  $s(u, j)/\tau$  in Eq. (15) reach very small values. Thus, when using Taylor expansion, the second- and higher-order terms can be ignored. Finally, because  $|N_u| \gg 1$  in Eq. (16), it is approximated by Eq. (17).

We explain how to derive CCL [25] from SSM [46]. It has two heuristic hyperparameters for the weight *w* of negative samples and the margin *M* for hard negative filtering. Assuming w = 1 and no negative filtering, CCL [25] is interpreted by  $\lim_{\tau \to +\infty} \mathcal{L}_{SSM}$ .

τ-

$$\lim_{d \to +\infty} \mathcal{L}_{SSM} \approx \mathcal{L}_{CCL}(w = 1) \text{ without hard negative filtering}$$
$$= \sum_{(u,i,\mathcal{N}_u) \in \mathcal{T}_s} \left( 1 - s(u,i) + \frac{1}{|\mathcal{N}_u|} \sum_{j \in \mathcal{N}_u} s(u,j) \right).$$
(18)

We then deal with the relationship between SSM [46] and BC [56]. While both loss functions employ the softmax loss function, BC also uses a margin. Assuming  $M_{ui} = 0$ , BC is induced to SSM. SSM does not adjust weight based on positive pairs, and it is difficult to train hard positive samples, which may degrade the model performance.

$$\mathcal{L}_{BC}(M_{ui} = 0) = \mathcal{L}_{SSM}$$
$$= -\sum_{(u,i,\mathcal{N}_u)\in\mathcal{T}_s} \log \frac{e^{s(u,i)/\tau}}{e^{s(u,i)/\tau} + \sum_{j\in\mathcal{N}_u} e^{s(u,j)/\tau}},$$
(19)

where  $M_{ui}$  is an interaction-wise margin for user u and item i.

#### 3.3 Analysis of Alignment and Uniformity

We derive the relationship between SSM [46] and DirectAU [37]. According to Theorem 1 in [40], InfoNCE, *i.e.*, SSM [46] is understood by optimizing alignment and uniformity. Specifically, for fixed  $\tau > 0$  and  $|\mathcal{N}_u| \rightarrow \infty$ , it is also proved that SSM asymptotically

converges to the combination of alignment and uniformity loss.

$$\lim_{|\mathcal{N}_{u}|\to\infty} - \mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} \left[ \log \frac{e^{s(u,i)/\tau}}{e^{s(u,i)/\tau} + \sum_{j\in\mathcal{N}_{u}} e^{s(u,j)/\tau}} \right] - \log |\mathcal{N}_{u}|$$
(20)

$$= -\frac{1}{\tau} \mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} s(u,i) + \mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} \left[ \lim_{|\mathcal{N}_{u}|\to\infty} \log\left(\frac{1}{|\mathcal{N}_{u}|}e^{s(u,i)/\tau} + \frac{1}{|\mathcal{N}_{u}|}\sum_{j\in\mathcal{N}_{u}}e^{s(u,j)/\tau}\right) \right]$$
(21)

$$= -\frac{1}{\tau} \mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} s(u,i) + \mathop{\mathbb{E}}_{u\sim p_{data}} \left[ \log \mathop{\mathbb{E}}_{j\sim p_{data}} e^{s(u,j)/\tau} \right]$$
(22)

$$\approx -\frac{1}{\tau} \mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} s(u,i)$$
  
+ 
$$\log \mathop{\mathbb{E}}_{(u,u')\sim p_{war}} e^{-2\cdot d(u,u')} + \log \mathop{\mathbb{E}}_{(i,i')\sim p_{war}} e^{-2\cdot d(i,i')}.$$
(23)

Here,  $p_{data}$  denotes the distribution of data. We derive Eq. (22) from Eq. (21) using the strong law of large numbers and the continuous mapping theorem. Then, the second term of Eq. (22) is divided by the user and item uniformity, using the logarithm of average pairwise Gaussian potential in [40].

Surprisingly, Eq. (23) is the equivalent form to DirectAU [37] using alignment and uniformity for recommendation. SSM [46] is asymptotically approximated by combining the alignment of positive pairs and the uniformity for users and items. Besides, the temperature  $\tau$  in Eq. (23) is interpreted as a hyperparameter to control the significance between alignment and uniformity, as used in  $\gamma$  in Eq (4). As  $\tau$  increases, it highlights the uniformity.

From the previous analysis of setwise and pairwise loss functions, most of them are distinguished by the negative sampling-related parameters (*i.e.*, temperature  $\tau$  and the number of negatives  $|N_u|$ ). Since DirectAU [37] is SSM [46] as  $|N_u|$  goes infinity, DirectAU can have the same effect as sampling many negative items in SSM. Thus, DirectAU achieves the effect of negative sampling without any sampling strategies, leading to better performance than SSM.

#### 4 PROPOSED LOSS FUNCTION

In this section, we propose a novel loss function by considering the unique characteristics of a target dataset, called *Margin-aware Alignment and Weighted Uniformity (MAWU)*. It takes the advantages of both BC [56] and DirectAU [37]. Our proposed MAWU is composed of two loss functions: (i) *margin-aware alignment (MA)* incorporates the margin into the alignment loss, and (ii) *weighted uniformity (WU)* controls the importance of users and items depending on user/item distributions.

**Margin-aware alignment loss.** Our analysis revealed that (i) BC [56] is interpreted as SSM [46] with the margin. (ii) DirectAU [37] directly learns user/item representations without negative sampling. Based on the relationship between DirectAU and SSM, we

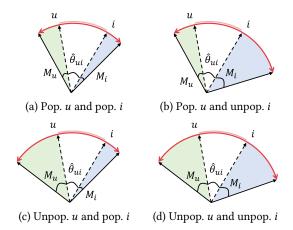


Figure 2: Four user *u*-item *i* cases in the training process for margin-aware alignment loss of MAWU. As training progresses, unpopular users and items have larger margin values than popular users and items. The green and blue backgrounds mean user and item margins, respectively.

first transform BC into a combination of alignment and uniformity.

$$\begin{split} \lim_{|\mathcal{N}_{u}| \to +\infty} &- \mathbb{E} \left[ \log \frac{e^{\cos\left(\hat{\theta}_{ui} + M_{ui}\right)/\tau}}{e^{\cos\left(\hat{\theta}_{ui} + M_{ui}\right)/\tau} + \sum_{j \in \mathcal{N}_{u}} e^{\cos\left(\hat{\theta}_{uj}\right)/\tau}} \right] \\ &- \log |\mathcal{N}_{u}| \qquad (24) \\ &\approx -\frac{1}{\tau} \mathbb{E} \cos\left(\hat{\theta}_{ui} + M_{ui}\right) \\ &+ \log \mathbb{E} e^{-2 \cdot d(u,u')} + \log \mathbb{E} e^{-2 \cdot d(i,i')}. \quad (25) \end{split}$$

Here,  $\hat{\theta}_{ui}$  means the angle between f(u) and f(i). The margin in BC [56] is used to discriminate the gap between positive pair (u, i) and negative pair (u, j), *i.e.*,  $\cos(\hat{\theta}_{ui} + M_{ui}) > \cos(\hat{\theta}_{uj})$ , penalizing the angular distance between user u and item i in positive pair (u, i). In the alignment loss, we also employ the margin to make a large loss value, stretching a user-item angular distance.

Specifically, we devise a new margin function that combines two margins for users and items. Because the margin varies depending on users and items, we represent the margin as a sum of a user margin and an item margin, *i.e.*,  $M_{ui} = M_u + M_i$ , where  $M_u \in \mathbb{R}^{|\mathcal{U}|}$  and  $M_i \in \mathbb{R}^{|\mathcal{I}|}$  are trainable parameters. While BC [56] additionally introduces a popularity encoder to estimate  $M_{ui} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ , we simply use  $|\mathcal{U}| + |\mathcal{I}|$  parameters without an additional module. In Appendix A, we show theoretically that MA loss function can learn more user/item-specific embeddings than BC [56].

Despite its simplicity, it can capture the user/item-specific margins and mitigate the biases from the active users and popular items. Figure 2 depicts four user-item cases in the training process based on user/item popularity. Particularly, it emphasizes positive pairs with unpopular users and items (See Figure 2(d)). It is effective for adaptively learning them with the margin depending on user and item popularity. Because the cosine function is non-monotonic, Seongmin Park, Mincheol Yoon, Jae-woong Lee, Hogun Park, & Jongwuk Lee

Table 2: Statistics of the datasets after preprocessing and the ratios of Gini indices between users (*i.e.*,  $Gini_u$ ) and items (*i.e.*,  $Gini_i$ ) for each dataset.

Dataset	#user	#item	#inter.	Density	Gini <sub>i</sub> /Gini <sub>u</sub>
Beauty	22.4k	12.1k	198.5k	0.07%	2.096
Gowalla	29.9k	41.0k	1,027.4k	0.08%	0.518
Yelp2018	31.7k	38.0k	1,561.4k	0.13%	1.090

we also restrict the range of the cosine function to make it monotonic, *i.e.*,  $0 \leq \hat{\theta}_{ui} + M_u + M_i \leq \pi$ . Formally, we formulate the margin-aware alignment loss function.

$$\mathcal{L}_{MA} = -\mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} \cos\left(\hat{\theta}_{ui} + M_u + M_i\right).$$
(26)

Weighted uniformity loss. Existing studies [37, 40] have used the same weights for user and item uniformity loss functions without considering the difference between users and items. However, in real-world recommendation datasets, the distribution of interactions can vary depending on users and items. If this perspective is neglected, it may lead to sub-optimal results. Table 2 provides the different ratios of Gini indices [5] between users and items in three datasets. Here,  $Gini_u$  and  $Gini_i$  represent the Gini indices that measure the distribution of interactions within the sets of users  $\mathcal{U}$  and items  $\mathcal{I}$ , respectively. A small  $Gini_u$  means that most users tend to rate items evenly. Therefore, it is necessary to highlight user uniformity if the dataset has more diversity in users than items, *i.e.*, the dataset has high  $Gini_i/Gini_u$ .

Based on these observations, we adopt different weights for user and item uniformities for each dataset. While this approach introduces additional hyperparameters, it enables us to more effectively address the statistical differences between users and items. Formally, we represent the weighted uniformity loss function as follows.

$$\mathcal{L}_{WU} = \gamma_1 \log \mathop{\mathbb{E}}_{(u,u') \sim p_{user}} e^{-2 \cdot d(u,u')} + \gamma_2 \log \mathop{\mathbb{E}}_{(i,i') \sim p_{item}} e^{-2 \cdot d(i,i')}$$
(27)

where  $\gamma_1$  and  $\gamma_2$  are the hyperparameters to control user and item uniformities, respectively.

**Training and Inference.** For model training, we lastly formulate the MAWU loss function that combines margin-aware alignment and weighted uniformity loss functions.

$$\mathcal{L}_{MAWU} = \mathcal{L}_{MA} + \mathcal{L}_{WU}.$$
 (28)

For inference, we make predictions using the inner product of user and item embedding vectors without normalization. Note that the user and item margins are only used for robust model training.

$$\hat{r}_{ui} = f(u)^{\top} f(i). \tag{29}$$

## **5 EXPERIMENTAL SETUP**

**Datasets and preprocessing**. We used three benchmark datasets, Beauty<sup>2</sup>, Gowalla<sup>3</sup>, and Yelp2018<sup>4</sup>, widely used in existing studies [13, 25, 37]. Following [37, 41], we use the 5-core setting for

<sup>&</sup>lt;sup>2</sup>https://jmcauley.ucsd.edu/data/amazon/links.html

<sup>&</sup>lt;sup>3</sup>http://snap.stanford.edu/data/loc-gowalla.html

<sup>&</sup>lt;sup>4</sup>https://www.yelp.com/dataset

CIKM '23, October 21-25, 2023, Birmingham, United Kingdom

Beauty and the 10-core setting for the other datasets. Also, we leave the last interacted item if item interactions are duplicated. Table 2 reports the statistics of the three datasets after preprocessing. We split each dataset into train:valid:test sets with a ratio of 7:1:2.

Baseline loss functions and CF models. We compare MAWU with ten CF loss functions in Table 1. To compare different loss functions, we used MF [17] and LightGCN [13] as the backbone model. Specifically, ten loss functions used in the evaluation are categorized into four groups: (i) Pointwise loss (i.e., BCE [17], MCL [11], and UIB [62]) (ii) Pairwise loss (i.e., BPR [29], CML [16], and SML [21]) (iii) Setwise loss (i.e., CCL [25], SSM [46], and BC Loss [56]) (iv) DirectAU [37]. To further verify the competitive edge of our loss function, it is compared with eight state-of-the-art CF models. We adopt three MF-based models (i.e., NeuMF [14], ENMF [2], and SimpleX [25]) and two AE-based models (i.e., MultVAE [23] and Rec-VAE [33]) and two GNN-based models (i.e., SGL [45], SimGCL [53], and LightGCL [1]). To verify the effectiveness of only the loss function, random sampling is used in MCL [11], and metadata (e.g., item features) is not used in CML [16]. Besides, UIB [62] leverages BPR [29] as a backbone loss function.

**Evaluation metrics**. We use two popular metrics, *Normalized Discounted Cumulative Gain* (*NDCG@N*) and *Recall@N*, to evaluate top-*N* recommendation. We use  $N = \{10, 20, 50\}$  for all datasets.

Reproducibility. All the methods are implemented in Recoole [58] framework and optimized using Adam [18] optimizer, and their parameters are initialized with Xavier's method [12]. For all CF models, we set latent dimension, learning rate, batch size, and the number of negative samples to 64, 0.001, 2,048, and 30, respectively. Besides, weight decay is tuned in {0, 1e-2, 1e-4, 1e-6, 1e-8}. For BC and SSM, negative samples are replaced with in-batch samples. The number of layers is 2 for all the LightGCN-based models. We set the max epoch to 1,000 and the early stopping epoch to 10 based on NDCG@20. For their own hyperparameters of each model, we carefully search within the optimum range based on their original papers. For the loss functions with a constant margin, we searched in the range from 0.2 to 1. For our loss, we explored  $\gamma_1$ ,  $\gamma_2$  in the range from 0.1 to 5 for all datasets. All the tables and figures that report only NDCG@N results also show similar trends for Recall@N results. All the results are averaged over 5 runs. The source code is available at https://github.com/psm1206/MAWU.

## 6 EXPERIMENTAL RESULTS

This section evaluates the MAWU loss function compared to the baseline loss functions and state-of-the-art CF models. We also validate the effectiveness of margin-aware alignment loss and weighted uniformity loss in MAWU.

## 6.1 Performance Comparison

**Evaluation for baseline loss functions**. Table 3 shows experimental results between ten existing loss functions and the MAWU loss function using two backbone models, *i.e.*, MF [17] and Light-GCN [13], on the three datasets. We report experimental results for each loss category as follows.

- Alignment and uniformity loss functions: In all the cases, MAWU outperforms DirectAU [37] on average by 3.13%, 2.11%, and 1.77%, respectively. This means that our proposed MAWU is more effective in improving recommendation performance compared to DirectAU. Interestingly, as a dataset becomes sparser (Yelp → Gowalla → Beauty), the performance gains of MAWU generally increase. This is because MA provides more learning opportunities for unpopular users and items in sparse datasets.
- Setwise loss functions: Except for the results in the Beauty-MF, either CCL [25] or BC [56], both of which utilize a margin, achieves the best performance. This indicates that incorporating a margin helps improve recommendation performance. Besides, the setwise loss functions (*i.e.*, CCL, SSM, and BC) consistently outperform the pairwise loss functions (*i.e.*, BPR, CML, and SML). This is because the setwise loss functions regard multiple negative items for a single positive item, unlike the pairwise loss functions that only consider a single negative item.
- Pairwise loss functions: In all cases, BPR [29] performs better than CML [16] and SML [21]. This indicates that -LogSigmoid is an effective activation function compared to ReLU. When comparing SML and CML, SML has performance improvements over CML of 29.38%, 18.87%, and 18.60% on average in Beauty, Gowalla, and Yelp2018 for MF backbone, respectively. This suggests that SML's adaptive margin and item-centric loss are used to achieve significant gains.
- **Pointwise loss functions**: We observed two interesting findings about the pointwise loss functions. (i) When LightGCN is the backbone model, UIB is superior to MCL with a constant margin. However, when using MF as the backbone, it shows the opposite trend. This is because the UIB margin is estimated with user embeddings extracted from LightGCN, which is richer than MF. (ii) The pointwise loss functions have on-par performance compared to the setwise loss functions. This is because we set the number of negative samples as high as 30. According to [25], BCE, one of the pointwise loss functions, performs worse than BPR when the number of negative samples is small while significantly outperforms BPR when the number of negative samples exceeds a certain number (10-20 in the paper).

**Evaluation for CF models**. Table 4 shows the performance of the proposed model (*i.e.*, MAWU-MF and MAWU-LightGCN) and state-of-the-art MF-, AE-, and GNN-based CF models. Our experimental results show that MF and LightGCN equipped with MAWU consistently outperform other CF models. This indicates that the proposed loss function MAWU leads to better performance, even when used with simple backbone models (*i.e.*, MF and LightGCN). Additionally, we found that GNN-based models (*i.e.*, SGL, SimGCL, and LightGCL) have better performance than MF-based models (*i.e.*, NeuMF, ENMF, and SimpleX) and AE-based models (*i.e.*, MultVAE and RecVAE). This is because GNN-based models utilize high-order relationships from a user-item graph.

## 6.2 Breakdown Analysis

All experiments in this section use LightGCN as a backbone model.

**Ablation study**. To validate the effectiveness of two components, *i.e.*, margin-aware alignment (MA) and weighted uniformity (WU),

 $<sup>^5</sup>$  On Gowalla-MF, CCL has salient performance because Gowalla has a relatively uniform distribution of items, and CCL highlights more uniformity ( $\tau$  goes to infinity).

Table 3: Accuracy comparison of ours and baselines on Beauty, Gowalla, and Yelp2018.  $\dagger$  indicates the best baseline, *i.e.*, DAU for t-test and performance gain. \* and \*\* indicate p < 0.05 and p < 0.01 for a one-tailed t-test. Bold indicates the best performance between MAWU and the best baseline, *i.e.*, DAU.  $\ddagger$  indicates higher performance than MAWU<sup>5</sup>.

		Backbone: MF				Backbone: LightGCN							
		Recall@N			1	NDCG@N			Recall@N		NDCG@N		
Dataset	Loss	N=10	N=20	N=50	N=10	N=20	N=50	N=10	N=20	N=50	N=10	N=20	N=50
	BCE [17]	0.0738	0.1065	0.1618	0.0450	0.0542	0.0666	0.0920	0.1304	0.1953	0.0555	0.0663	0.0808
	MCL [11]	0.0922	0.1304	0.1944	0.0569	0.0676	0.0819	0.0892	0.1284	0.1947	0.0535	0.0644	0.0793
	UIB [62]	0.0829	0.1189	0.1769	0.0499	0.0600	0.0731	0.0895	0.1270	0.1883	0.0541	0.0647	0.0784
-	BPR [29]	0.0885	0.1271	0.1944	0.0533	0.0641	0.0791	0.0882	0.1283	0.1992	0.0521	0.0632	0.0789
	CML [16]	0.0545	0.0828	0.1326	0.0329	0.0410	0.0523	0.0328	0.0492	0.0817	0.0202	0.0249	0.0325
Beauty	SML [21]	0.0712	0.1055	0.1645	0.0422	0.0518	0.0652	0.0748	0.1119	0.1756	0.0442	0.0545	0.0686
Deauty	CCL [25]	0.0751	0.1036	0.1489	0.0483	0.0563	0.0664	0.0716	0.0980	0.1394	0.0451	0.0524	0.0617
	SSM [46]	0.0891	0.1314	0.1986	0.0537	0.0655	0.0805	0.0870	0.1261	0.1958	0.0517	0.0627	0.0782
	BC [56]	0.0877	0.1274	0.1911	0.0527	0.0639	0.0782	0.0874	0.1276	0.1961	0.0522	0.0633	0.0786
-	DAU <sup>†</sup> [37]	0.0975	0.1399	<b>0.2064</b> <sup>‡</sup>	0.0598	0.0716	0.0865	0.0983	0.1412	0.2134	0.0592	0.0711	0.0872
	MAWU (ours)	0.1004*	0.1405	0.2036	0.0628**	0.0740**	0.0881*	0.1030*	0.1457*	0.2135	0.0635**	0.0754**	0.0905**
	Gain (%)	3.02	0.40	-1.34	5.05	3.35	1.90	4.74	3.16	0.07	7.30	6.10	3.83
	BCE [17]	0.1181	0.1731	0.2763	0.1132	0.1302	0.1614	0.1343	0.1938	0.3027	0.1307	0.1482	0.1807
	MCL [11]	0.1319	$0.1946^{\ddagger}$	0.3085 <sup>‡</sup>	0.1242	0.1433	0.1775	0.1122	0.1635	0.2628	0.1092	0.1245	0.1543
	UIB [62]	0.1177	0.1755	0.2838	0.1112	0.1293	0.1619	0.1295	0.1877	0.2952	0.1251	0.1425	0.1747
	BPR [29]	0.1153	0.1692	0.2713	0.1084	0.1255	0.1562	0.1284	0.1866	0.2952	0.1247	0.1422	0.1747
	CML [16]	0.0590	0.0987	0.1848	0.0519	0.0652	0.0915	0.0485	0.0810	0.1515	0.0430	0.0538	0.0752
Gowalla	SML [21]	0.0710	0.1169	0.2130	0.0623	0.0775	0.1066	0.0559	0.0902	0.1611	0.0458	0.0580	0.0800
	CCL [25]	$0.1394^{\ddagger}$	0.1993 <sup>‡</sup>	0.3003	$0.1377^{\ddagger}$	$0.1543^{\ddagger}$	0.1838 <sup>‡</sup>	0.1284	0.1847	0.2794	0.1264	0.1422	0.1698
	SSM [46]	0.1215	0.1819	0.2940	0.1137	0.1324	0.1661	0.1093	0.1601	0.2609	0.1059	0.1212	0.1514
	BC [56]	0.1194	0.1821	0.3007	0.1104	0.1300	0.1655	0.1094	0.1603	0.2605	0.1061	0.1213	0.1514
	DAU <sup>†</sup> [37]	0.1293	0.1881	0.2979	0.1257	0.1431	0.1758	0.1313	0.1905	0.3010	0.1287	0.1461	0.1789
	MAWU (ours)	0.1326**	0.1921**	0.3022**	0.1277**	0.1454**	0.1783**	0.1351**	0.1956**	0.3074**	0.1317**	0.1496**	0.1829**
	Gain (%)	2.55	2.08	1.45	1.58	1.59	1.40	2.86	2.68	2.11	2.35	2.41	2.24
	BCE [17]	0.0532	0.0879	0.1615	0.0606	0.0723	0.0984	0.0594	0.0973	0.1772	0.0691	0.0814	0.1094
	MCL [11]	0.0608	0.1000	0.1819	0.0705	0.0831	0.1118	0.0542	0.0895	0.1647	0.0624	0.0742	0.1007
	UIB [62]	0.0468	0.0791	0.1500	0.0535	0.0644	0.0897	0.0582	0.0956	0.1745	0.0669	0.0792	0.1070
	BPR [29]	0.0434	0.0729	0.1384	0.0486	0.0588	0.0820	0.0564	0.0921	0.1693	0.0647	0.0764	0.1035
Yelp2018 -	CML [16]	0.0292	0.0519	0.1070	0.0333	0.0414	0.0612	0.0400	0.0698	0.1379	0.0455	0.0558	0.0801
	SML [21]	0.0352	0.0614	0.1227	0.0399	0.0491	0.0709	0.0295	0.0524	0.1075	0.0334	0.0415	0.0614
	CCL [25]	0.0521	0.0823	0.1415	0.0614	0.0706	0.0910	0.0522	0.0831	0.1459	0.0613	0.0708	0.0927
	SSM [46]	0.0566	0.0937	0.1722	0.0657	0.0777	0.1052	0.0486	0.0809	0.1513	0.0564	0.0671	0.0919
	BC [56]	0.0606	0.1010	0.1856	0.0699	0.0831	0.1128	0.0492	0.0815	0.1529	0.0568	0.0676	0.0928
-	DAU <sup>†</sup> [37]	0.0639	0.1031	0.1834	0.0745	0.0870	0.1151	0.0658	0.1054	0.1872	0.0773	0.0897	0.1182
-	MAWU (ours)	0.0656**	0.1058**	0.1871**	0.0766**	0.0894**	0.1178**	0.0664	0.1066**	0.1895**	0.0779*	0.0905*	0.1194**
	Gain (%)	2.66	2.62	2.02	2.82	2.76	2.35	0.85	1.14	1.25	0.80	0.89	1.00

we evaluate the performance of ablating MA and WU from MAWU-LightGCN on Beauty, Gowalla, and Yelp2018. Table 5 shows the results of the ablation study. Here, '-MA' and '-WU' are the removal of MA and WU from MAWU, respectively. We can see that eliminating MA and WU leads to a performance drop, indicating that both are effective. Interestingly, on Gowalla and Yelp2018, we found that the performance of MA and WU decreased slightly when only one of them was removed. This suggests that using only one of MA and WU is effective in learning better user and item representations.

**Comparison with various margin types in MA**. We evaluate the five margin types. The first two types have a constant margin, and the other types learn the margin using UIB [62], BC [56], and our fashion (Eq. (26)), respectively.

- (1) Zero margin:  $M_u$  and  $M_i = 0$ , *i.e.*, DirectAU [37].
- (2) Inverse popularity:  $M_u = \frac{p_{u,\max}}{p_u}$  and  $M_i = \frac{p_{i,\max}}{p_i}$ , where  $p_u, p_i, p_{u,\max}$ , and  $p_{i,\max}$  are user u, item i popularity, max popularity of users and items, respectively.
- (3) UIB fashion [62]:  $M_u = W^T f(u)$  and  $M_i = W^T f(i)$ , where  $W \in \mathbb{R}^{d \times 1}$  is a learnable parameter.
- (4) BC fashion [56]: Its margin is determined as an angle between user and item popularity embedding.
- (5) Ours: It learns the margin using Eq. (26).

Table 6 shows the highest performance of the type (5) we used. Notably, compared to type (1), types (2) and (4) lead to performance improvement. This indicates that the popularity information helps determine the margin. Type (4) shows better performance than

Beauty, Gowalla, and Yelp2018.

Table 4: Accuracy comparison of ours and CF models on

N@10 N@20 N@50 Dataset Model NeuMF [14] 0.0373 0.0455 0.0570 ENMF [2] 0.0522 0.0756 0.0621 SimpleX [25] 0.0542 0.0658 0.0813 MultVAE [23] 0.0438 0.0526 0.0646RecVAE [33] 0.0518 0.0615 0.0739 Beauty SGL [45] 0.0556 0.0822 0.0670 SimGCL [53] 0.0562 0.0657 0.0773 LightGCL [1] 0.0575 0.0688 0.0835 MAWU-MF 0.0628 0.0740 0.0881 MAWU-LightGCN 0.0635 0.0905 0.0754 NeuMF [14] 0.0884 0.1017 0.1277 ENMF [2] 0.1055 0.1199 0.1475 SimpleX [25] 0.1179 0.1363 0.1699 MultVAE [23] 0.1584 0.1281 0.1115 Gowalla RecVAE [33] 0.1195 0.1365 0.1672 0.1772 SGL [45] 0.1281 0.1452 SimGCL [53] 0.1226 0.1395 0.1703 LightGCL [1] 0.1236 0.1417 0.1751 MAWU-MF 0.1277 0.1454 0.1783 MAWU-LightGCN 0.1317 0.1496 0.1829 NeuMF [14] 0.0417 0.0509 0.0718 ENMF [2] 0.0658 0.0768 0.1027 SimpleX [25] 0.0683 0.0794 0.1047 MultVAE [23] 0.0594 0.0714 0.0977 0.0689 RecVAE [33] 0.0809 0.1080 Yelp2018 SGL [45] 0.0668 0.0788 0.1062 SimGCL [53] 0.0732 0.0848 0.1111 LightGCL [1] 0.0688 0.0803 0.1074 MAWU-MF 0 1178 0 0766 0.0894 MAWU-LightGCN 0.0779 0.0905 0.1194

Table 5: Abalation study on Beauty, Gowalla, and Yelp2018.

001104140		on Douary,		
Dataset	Model	N@10	N@20	N@50
	MAWU	0.0635	0.0754	0.0905
Beauty	-MA	0.0624	0.0738	0.0883
,	-WU	0.0628	0.0743	0.0887
	MAWU	0.1317	0.1496	0.1829
Gowalla	-MA	0.1315	0.1491	0.1827
	-WU	0.1310	0.1482	0.1820
	MAWU	0.0779	0.0905	0.1194
Yelp2018	-MA	0.0778	0.0903	0.1193
-	-WU	0.0773	0.0898	0.1185

Table 6: Accuracy comparison with margin types on Beauty.

Model	N@10	N@20	N@50
(1) Zero margin	0.0592	0.0711	0.0872
(2) Inverse popularity	0.0608	0.0720	0.0865
(3) UIB fashion [62]	0.0606	0.0720	0.0867
(4) BC fashion [56]	0.0615	0.0731	0.0871
(5) Ours	0.0635	0.0754	0.0905

type (2) because it learns the margin. Without explicitly using the popularity information, types (3) and (5) also result in improved performance. Type (3) has a considerable variation according to the backbone model because the margin is obtained by user embedding. However, type (5) learns stably using separate layers and has high performance and fewer parameters than other types. Thus, we adopt type (5) as the margin learning method for the proposed MA.

CIKM '23, October 21-25, 2023, Birmingham, United Kingdom

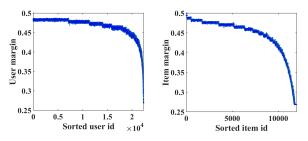


Figure 3: Correlation between user/item margin, *i.e.*,  $M_u$  and  $M_i$ , and user/item id sorted by popularity on Beauty. The larger the user/item id, the higher user/item popularity.

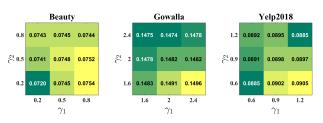


Figure 4: Heatmaps of hyperparameters  $\gamma_1$  and  $\gamma_2$  in Eq. (27) on Beauty, Gowalla, and Yelp2018. Each metric is NDCG@20.

**Margin analysis for user/item popularity**. Figure 3 depicts that user and item margins learned from MAWU for each user and item sorted by popularity. As we intended, the unpopular users and items have larger margin values. That is, the margin encourages learning for unpopular users and items. Besides, the user and item margins have various margin values within the range of 0.2693-0.4861 and 0.2688-0.4948, respectively. This means that there are suitable margin values for each user and item.

**Effect of hyperparameters**  $\gamma_1$  and  $\gamma_2$  in WU. Figure 4 presents the results of a grid search for  $\gamma_1$  and  $\gamma_2$  of the MAWU loss on three different datasets. Lighter colors indicate higher performance in terms of NDCG@20. Our experiments reveal intriguing findings: (i) The performance distribution varies across datasets. In particular, the ratio  $\gamma_1/\gamma_2$  of the best combinations for Beauty, Gowalla, and Yelp2018 are 4, 1.5, and 2, respectively, which is proportional to the ratio of Gini indices in Table 2. This means that if the item distribution is skewed (large *Gini*<sub>i</sub>), the weight for item uniformity should be lowered. (ii) Among all datasets, the highest performance is consistently achieved when  $\gamma_1$  is greater than  $\gamma_2$ . It indicates that achieving uniformity in user representations is more critical than in item representations. From these findings, it is necessary to adjust  $\gamma_1$  and  $\gamma_2$  properly according to the characteristics of the dataset.

## 7 RELATED WORK

This section reviews existing studies for two parts, *i.e.*, interaction encoder and negative sampling, used in CF models because loss functions are addressed in the previous sections (Refer to Appendix C for related work on loss functions).

**Interaction encoder.** A lot of existing studies have been actively studied to develop better interaction encoders. It is broadly categorized into two directions: linear and non-linear models. (i) The

Seongmin Park, Mincheol Yoon, Jae-woong Lee, Hogun Park, & Jongwuk Lee

linear model represents the latent embedding vectors of users and items as a linear combination. It includes neighbor-based regression [26, 34] and matrix factorization [2, 19, 31]. Despite its simplicity, recent studies [5, 6, 9] have shown that linear models often outperform non-linear models. (ii) Non-linear models mostly utilize deep neural networks, such as autoencoders (AE), recurrent neural networks (RNN), graph neural networks (GNN), and so on. Specifically, AE-based models [23, 32, 33] learn complex non-linear relationships among items. RNN-based models [15, 24, 43] adopt the user's sequential behavior information. They are beneficial for sequential and session-based recommender systems. GNN-based models [13, 45, 53] capture high-order correlations from a user-item bipartite graph. They learn multi-hop informative relationships, leading to better performances than existing CF models.

**Negative sampling.** Most CF models utilize implicit feedback, *e.g.*, click and view, which is more applicable to real-world scenarios. So, we need to treat some of the implicit feedback as negative feedback, which is indispensable for pairwise and setwise loss functions. As a sampling method, random sampling [29] is widely used, but several studies focus on using more informative negative sampling [4, 7, 8, 28, 44, 50, 54, 55, 57]. Specifically, [4, 44] replaced the uniform distribution with an item popularity-based distribution, and [28, 57] gave a high sampling probability to negative samples with large prediction scores. Based on the observation that false negative samples have high prediction scores, [8] chooses negative items.

## 8 CONCLUSION

This paper is the first work that mathematically/empirically analyzes existing loss functions. (i) Based on our analysis, we found the potential for model improvement through the integration of BC and DirectAU. (ii) From the potential, we proposed a new loss, called *Margin-aware Alignment and Weighted Uniformity (MAWU)*. In our proposed MAWU, MA encourages CF models to better learn user/item representations by using adaptive margins based on popularity. Then, WU adjusts the importance of user and item uniformity based on dataset statistics, *e.g.*, skewed/uniform distributions. (iii) Extensive experimental results demonstrate that LightGCN equipped with MAWU outperforms existing loss functions and state-of-the-art MF-, AE-, and GNN-based CF models.

#### ACKNOWLEDGMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019-0-00421, 2022-0-00680, 2022-0-01045, RS-2023-00219919, and NRF-2018R1A5A1060031).

#### REFERENCES

- Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In ICLR.
- [2] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. ACM Trans. Inf. Syst. 38, 2 (2020), 14:1–14:28.
- [3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, Vol. 119. 1597–1607.
- [4] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On Sampling Strategies for Neural Network-based Collaborative Filtering. In KDD. 767–776.
- [5] Jin Yao Chin, Yile Chen, and Gao Cong. 2022. The Datasets Dilemma: How Much Do We Really Know About Recommendation Datasets?. In WSDM. 141–149.
- [6] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *RecSys.* 101–109.
- [7] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *IJCAI*. 2230–2236.
- [8] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. In *NeurIPS*.
- [9] Yushun Dong, Jundong Li, and Tobias Schnabel. 2023. When Newer is Not Better: Does Deep Learning Really Benefit Recommendation From Implicit Feedback?. In *SIGIR*.
- [10] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In EMNLP. 6894–6910.
- [11] Zhaolin Gao, Zhaoyue Cheng, Felipe Pérez, Jianing Sun, and Maksims Volkovs. 2022. MCL: Mixed-Centric Loss for Collaborative Filtering. In WWW. 2339–2347.
- [12] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, Vol. 9. 249–256.
- [13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In SIGIR. 639–648.
- [14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In WWW. 173–182.
- [15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.
- [16] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In WWW. 193–201.
- [17] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. 263–272.
- [18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
- [19] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In KDD. 426–434.
- [20] Chuming Li, Xin Yuan, Chen Lin, Minghao Guo, Wei Wu, Junjie Yan, and Wanli Ouyang. 2019. AM-LFS: AutoML for Loss Function Search. In ICCV. 8409–8418.
- [21] Mingming Li, Shuai Zhang, Fuqing Zhu, Wanhui Qian, Liangjun Zang, Jizhong Han, and Songlin Hu. 2020. Symmetric Metric Learning with Adaptive Margin for Recommendation. In AAAI. 4634–4641.
- [22] Zelong Li, Jianchao Ji, Yingqiang Ge, and Yongfeng Zhang. 2022. AutoLossGen: Automatic Loss Function Generation for Recommender Systems. In SIGIR. 1304– 1315.
- [23] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In WWW. 689–698.
- [24] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2017. A Deep Recurrent Collaborative Filtering Framework for Venue Recommendation. In CIKM. 1429– 1438.
- [25] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In CIKM. 1243–1252.
- [26] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *ICDM*. 497–506.
- [27] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-Class Collaborative Filtering. In *ICDM*. 502– 511.
- [28] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In WSDM. 273–282.
- [29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In UAI. 452– 461.
- [30] Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2015. Recommender Systems Handbook. Springer.
- [31] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In NIPS. 1257–1264.

CIKM '23, October 21-25, 2023, Birmingham, United Kingdom

- [32] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In WWW. 111–112.
- [33] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I. Nikolenko. 2020. RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. In WSDM. 528–536.
- [34] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In WWW. 3251-3257.
- [35] Yongxiang Tang, Wentao Bai, Guilin Li, Xialong Liu, and Yu Zhang. 2022. CROLoss: Towards a Customizable Loss for Retrieval Models in Recommender Systems. In CIKM. 1916–1924.
- [36] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. CoRR abs/1807.03748 (2018). arXiv:1807.03748
- [37] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards Representation Alignment and Uniformity in Collaborative Filtering. In KDD. 1816–1825.
- [38] Feng Wang and Huaping Liu. 2021. Understanding the Behaviour of Contrastive Loss. In CVPR. 2495–2504.
- [39] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In KDD. 1235–1244.
- [40] Tongzhou Wang and Phillip Isola. 2020. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML*, Vol. 119. 9929–9939.
- [41] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In SIGIR. 165–174.
- [42] Xiaobo Wang, Shuo Wang, Cheng Chi, Shifeng Zhang, and Tao Mei. 2020. Loss Function Search for Face Recognition. In *ICML*. 10029–10038.
- [43] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In WSDM. 495–503.
- [44] Ga Wu, Maksims Volkovs, Chee Loong Soon, Scott Sanner, and Himanshu Rai. 2019. Noise Contrastive Estimation for One-Class Collaborative Filtering. In SIGIR. 135–144.
- [45] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. In SIGIR. 726–735.
- [46] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2022. On the Effectiveness of Sampled Softmax Loss for Item Recommendation. *CoRR* abs/2201.02327 (2022).
- [47] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In WSDM. 153–162.

- [48] Haowen Xu, Hao Zhang, Zhiting Hu, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2019. AutoLoss: Learning Discrete Schedule for Alternate Optimization. In *ICLR*.
- [49] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*. 3203–3209.
- [50] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H. Chi. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In WWW. 441–447.
- [51] Menglin Yang, Zhihao Li, Min Zhou, Jiahong Liu, and Irwin King. 2022. HICF: Hyperbolic Informative Collaborative Filtering. In KDD. 2212–2221.
- [52] Menglin Yang, Min Zhou, Jiahong Liu, Defu Lian, and Irwin King. 2022. HRCF: Enhancing Collaborative Filtering via Hyperbolic Geometric Regularization. In WWW. 2462–2471.
- [53] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation. In SIGIR. 1294–1303.
- [54] Lu Yu, Shichao Pei, Feng Zhu, Longfei Li, Jun Zhou, Chuxu Zhang, and Xiangliang Zhang. 2022. A Biased Sampling Method for Imbalanced Personalized Ranking. In CIKM. 2393–2402.
- [55] Wenhui Yu and Zheng Qin. 2020. Sampler Design for Implicit Feedback Data by Noisy-label Robust Learning. In SIGIR. 861–870.
- [56] An Zhang, Wenchang Ma, Xiang Wang, and Tat seng Chua. 2022. Incorporating Bias-aware Margins into Contrastive Loss for Collaborative Filtering. In *NeurIPS*.
- [57] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In SIGIR. 785–788.
- [58] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In CIKM. 4653–4664.
- [59] Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, and Chong Wang. 2021. AutoLoss: Automated Loss Function Search in Recommendations. In KDD. 3959–3967.
- [60] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems. In KDD. 3985–3995.
- [61] Ziwei Zhu, Jianling Wang, and James Caverlee. 2019. Improving Top-K Recommendation via Joint Collaborative Autoencoders. In WWW. 3483–3482.
- [62] Jianhuan Zhuo, Qiannan Zhu, Yinliang Yue, and Yuhong Zhao. 2022. Learning Explicit User Interest Boundary for Recommendation. In WWW. 193–202.

Seongmin Park, Mincheol Yoon, Jae-woong Lee, Hogun Park, & Jongwuk Lee

Table 7: Accuracy comparison in two user groups (2:8) by popularity. Pop and Unpop mean popular and unpopular user groups, respectively. DAU and MAWU use MF as a backbone. Although we only report NDCG@N results, we observe similar trends with Recall@N.

$\begin{array}{c c c c c c c c c c c c c c c c c c c $						
$\begin{array}{c} & \begin{array}{c} & & & & & & & & & & & & & & & & & & &$	Dataset	Group	Loss	N@10	N@20	N@50
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$			DAU	0.0859	0.1047	0.1309
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		Pop	MAWU	0.0952	0.1121	0.1367
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	Roonty		Gain (%)	10.83	7.07	4.43
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	beauty –		DAU	0.0534	0.0633	0.0750
$ \begin{array}{c} & DAU & 0.1798 & 0.1766 & 0.20 \\ \hline DAU & 0.1798 & 0.1766 & 0.20 \\ \hline MAWU & 0.1810 & 0.1783 & 0.20 \\ \hline Gain (\%) & 0.67 & 0.96 & 0.6 \\ \hline DAU & 0.1124 & 0.1349 & 0.10 \\ \hline Unpop & MAWU & 0.1146 & 0.1374 & 0.17 \\ \hline Gain (\%) & 1.96 & 1.85 & 1.6 \\ \hline DAU & 0.1284 & 0.1200 & 0.14 \\ \hline Pop & MAWU & 0.1317 & 0.1229 & 0.14 \\ \hline Gain (\%) & 2.57 & 2.42 & 2.1 \\ \hline \end{array} $		Unpop	MAWU	0.0555	0.0649	0.0766
$ \begin{array}{c} \mbox{Gowalla} & \begin{tabular}{ c c c c c c c c c c c c c c c c c c c$			Gain (%)	3.93	2.53	2.13
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$			DAU	0.1798	0.1766	0.2086
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	Gowalla –	Pop	MAWU	0.1810	0.1783	0.2099
DAU         0.1124         0.1349         0.16           Unpop         MAWU         0.1146         0.1374         0.17           Gain (%)         1.96         1.85         1.6           DAU         0.1284         0.1200         0.14           Pop         MAWU         0.1317         0.1229         0.14           Gain (%)         2.57         2.42         2.1			Gain (%)	0.67	0.96	0.62
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$			DAU	0.1124	0.1349	0.1683
DAU         0.1284         0.1200         0.14           Pop         MAWU         0.1317         0.1229         0.14           Gain (%)         2.57         2.42         2.1		Unpop	MAWU	0.1146	0.1374	0.1710
Pop MAWU 0.1317 0.1229 0.14 Gain (%) 2.57 2.42 2.1			Gain (%)	1.96	1.85	1.60
$\frac{1}{1} \frac{1}{1} \frac{1}$			DAU	0.1284	0.1200	0.1455
Gain (%) 2.57 2.42 2.1	Yelp2018 –	Pop	MAWU	0.1317	0.1229	0.1486
			Gain (%)	2.57	2.42	2.13
DAU 0.0609 0.0786 0.10			DAU	0.0609	0.0786	0.1078
Unpop MAWU 0.0623 0.0806 0.10		Unpop	MAWU	0.0623	0.0806	0.1096
Gain (%) 2.30 2.54 1.6			Gain (%)	2.30	2.54	1.67

## A MINIMIZATION OF MARGIN-AWARE ALIGNMENT LOSS FUNCTION

We theoretically prove that the MA loss function that expresses  $M_{ui}$  as the sum of  $M_u$  and  $M_i$  can learn more user/item-specific embeddings than BC [56] loss function.

THEOREM A.1. Assuming i) the user/item embeddings are normalized, and ii) there are rotation matrices that rotate the user/item embeddings, the minimization of MA loss function is equivalent to minimizing a compactness part. The compactness makes user/item embeddings closer to the averaged item/user embeddings that reflect user/item-specific information (e.g., user/item popularitiess).

$$\mathcal{L}_{MA} \stackrel{c}{=} \sum_{u \in \mathcal{U}} \left\| v_u - R_u^{\mathrm{T}} c_u^R \right\|^2 + \sum_{i \in I} \left\| v_i - R_i^{\mathrm{T}} c_i^R \right\|^2, \qquad (30)$$

where  $\stackrel{c}{=}$  denotes equality up to multiplicative or additive constant.  $c_u^R = \frac{1}{|\mathcal{P}_u|} \sum_{i \in \mathcal{P}_u} R_i v_i$  and  $c_i^R = \frac{1}{|\mathcal{P}_i|} \sum_{u \in \mathcal{P}_i} R_u v_u$  denote average of rotated item and user embeddings, respectively.  $\mathcal{P}_u$  and  $\mathcal{P}_i$  are a set of items by user u and a set of users by item i, respectively.  $R_u$  and  $R_i$  are rotation matrices of user and item embeddings with the margins  $M_u$  and  $M_i$ , respectively. Besides,  $v_u$  and  $v_i$  are L2-normalized user and item embedding, respectively <sup>6</sup>.

**PROOF.** To prove Theorem A.1, we are inspired by a theoretical property of BC [56] loss function. The theoretical property means that minimizing BC optimizes both compactness and dispersion simultaneously.

$$\mathcal{L}_{BC} \ge \sum_{u \in \mathcal{U}} \|v_u - c_u\|^2 + \sum_{i \in I} \|v_i - c_i\|^2 - \sum_{u \in \mathcal{U}} \sum_{j \in \mathcal{N}_u} \|v_u - c_j\|^2,$$
(31)

where  $c_u = \frac{1}{|\mathcal{P}_u|} \sum_{i \in \mathcal{P}_u} v_i$  and  $c_i = \frac{1}{|\mathcal{P}_i|} \sum_{u \in \mathcal{P}_i} v_u$  denote averaged item and user embeddings, respectively.

The first two terms are: i) compactness, which makes the user embedding close to the average of the item embeddings, and the last term is: ii) dispersion, which makes the distance between users and items farther apart. Since dispersion implies uniformity and is the corresponding concept of the WU loss function, we aim to verify the compactness of the MA loss function by referring to the first two terms.

Suppose we have rotation matrices (*i.e.*,  $R_u$  and  $R_i$ ) that rotate the user/item embeddings by the user/item margins (*i.e.*,  $M_u$  and  $M_i$ ), then the MA loss function is derived as follows.

$$\mathcal{L}_{MA} = -\sum_{(u,i)\in\mathcal{D}} \cos\left(\hat{\theta}_{ui} + M_u + M_i\right) = -\sum_{(u,i)\in\mathcal{D}} \left(R_u v_u\right)^{\mathrm{T}} \left(R_i v_i\right).$$
(32)

Here,  $v_u$  and  $v_i$  are unit vectors (*i.e.*,  $v_u^T v_i = 1 - \frac{1}{2} ||v_u - v_i||^2$ ), which can be expanded as follows.

$$-\sum_{(u,i)\in\mathcal{D}} (R_u v_u)^{\mathrm{T}} (R_i v_i) \stackrel{\mathrm{c}}{=} \sum_{(u,i)\in\mathcal{D}} \|R_u v_u - R_i v_i\|^2$$
(33)

$$= \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{P}_u} \left( \|R_u v_u\|^2 - v_u^{\mathrm{T}} R_u^{\mathrm{T}} R_i v_i \right) + \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{P}_i} \left( \|R_i v_i\|^2 - v_u^{\mathrm{T}} R_u^{\mathrm{T}} R_i v_i \right)$$
(34)

$$= \sum_{u \in \mathcal{U}} \left( |\mathcal{P}_{u}| v_{u}^{\mathrm{T}} R_{u}^{\mathrm{T}} R_{u} v_{u} - v_{u}^{\mathrm{T}} R_{u}^{\mathrm{T}} \sum_{i \in \mathcal{P}_{u}} R_{i} v_{i} \right) + \sum_{i \in I} \left( |\mathcal{P}_{i}| v_{i}^{\mathrm{T}} R_{i}^{\mathrm{T}} R_{i} v_{i} - v_{i}^{\mathrm{T}} R_{i}^{\mathrm{T}} \sum_{u \in \mathcal{P}_{i}} R_{u} v_{u} \right)$$
(35)

$$= \sum_{u \in \mathcal{U}} \left( |\mathcal{P}_{u}| v_{u}^{\mathrm{T}} \left( v_{u} - \frac{R_{u}^{\mathrm{T}}}{|\mathcal{P}_{u}|} \sum_{i \in \mathcal{P}_{u}} R_{i} v_{i} \right) \right) + \sum_{i \in I} \left( |\mathcal{P}_{i}| v_{i}^{\mathrm{T}} \left( v_{i} - \frac{R_{i}^{\mathrm{T}}}{|\mathcal{P}_{i}|} \sum_{u \in \mathcal{P}_{i}} R_{u} v_{u} \right) \right)$$
(36)

$$= \sum_{u \in \mathcal{U}} \left( |\mathcal{P}_u| \, v_u^{\mathrm{T}} \left( v_u - R_u^{\mathrm{T}} c_u^R \right) \right) + \sum_{i \in I} \left( |\mathcal{P}_i| \, v_i^{\mathrm{T}} \left( v_i - R_i^{\mathrm{T}} c_i^R \right) \right)$$
(37)

$$\stackrel{c}{=} \sum_{u \in \mathcal{U}} \left\| v_u - R_u^{\mathrm{T}} c_u^R \right\|^2 + \sum_{i \in \mathcal{I}} \left\| v_i - R_i^{\mathrm{T}} c_i^R \right\|^2.$$
(38)

From Eq (38), we can see that the MA loss function makes user embeddings closer to the average of item embeddings reflecting user-specific information (*e.g.*, user popularity) (and vice versa). This suggests that the MA loss function may yield more meaningful embeddings because it can reflect user-specific information compared to simply making user embeddings closer to the average of item embeddings (*i.e.*, BC [56]). In fact, Table 6 shows that the MA loss function method (*i.e.*, type (5)) outperforms the BC method (*i.e.*, type (4)).

<sup>&</sup>lt;sup>6</sup>For visibility of formulas, we denote  $v_u$  and  $v_i$  instead of f(u) and f(i), respectively.

## B PERFORMANCE ANALYSIS BY USER POPULARITY

To analyze where the performance improvement of MAWU comes from, we divide the user groups 2:8 by user popularity to observe the performance. Table 7 shows that MAWU outperforms DirectAU [37] in both groups. This seems to depend on the characteristics of the dataset.

# C RELATED WORK ON VARIOUS LOSS FUNCTIONS IN COLLABORATIVE FILTERING

**Overview.** Traditional loss functions, *i.e.*, MSE [19], BCE [17], and BPR [29], that have been widely used in recommender systems. While many SOTA models still enjoy using these loss functions, a variety of loss functions have been recently proposed in CF [11, 16, 21, 25, 37, 46, 62]. These studies are gaining more attention as they show that simply changing the loss function can outperform SOTA models [11, 25, 37, 46]. The research on CF loss function has been developed from the perspective of representation learning, from metric learning [16, 21] to contrastive learning [45, 46, 53, 56]. Recently, the decomposition of contrastive loss into alignment and uniformity has shown high performance [37]. In addition, module-type loss function [11, 62] and representation learning in hyperbolic space [51, 52] have been proposed. Furthermore, we formally compare the loss functions of graph contrastive learning-based models and traditional models.

**Metric learning.** Hsieh et al. [16] pioneered the development of the loss function, which first introduced the concept of metric learning to CF. CML [16] incorporates the intuition that the distance between a user and a positive item should be shorter than the distance between a user and a negative item into the loss function. While CML utilizes only user-centric loss, SML [21] considers the concept of item-centric loss additionally. Besides, unlike CML, which uses the margin constant, SML [21] adaptively learns the margin.

**Contrastive learning.** The loss functions that are interpreted in terms of contrastive learning have been proposed [25, 45, 46, 53, 56, 60]. A simple form of contrastive loss, CCL [25], performs hard negative sampling with a constant margin. In addition, InfoNCE [36] and NT-Xent [3] loss functions, which are popular contrastive loss functions in other domains (*e.g.*, computer vision and natural language processing), are also utilized in recommender systems [45, 46, 53, 56, 60]. In particular, Wu et al. [46] proves that the contrastive loss function mitigates popularity bias and has a hard negative sampling effect, showing its suitability with recommender systems. Motivated by the proof, Zhang et al. [56] presents a new loss function that adds an angular margin to SSM [46]. This margin limits the space where user embeddings can be learned by the angle formed by the popularity embedding of the user and the positive item.

Alignment and uniformity. Recently, there have been efforts to interpret contrastive loss as two new properties [37, 40]. Wang and

Isola [40] verified that minimizing contrastive loss has the same effect as optimizing alignment and uniformity. Inspired by the proof, Wang et al. [37] derives that the BPR loss is smaller than the vanilla BPR loss under perfect alignment and uniformity conditions. Based on the derivation, Wang et al. [37] proposed DirectAU [37], which directly optimizes alignment and uniformity in CF for the first time. DirectAU has the advantage of not requiring negative sampling.

**Modularization.** The module-type loss functions [11, 62] have been proposed that can be added to the existing loss function. Specifically, MCL [11] adds several weighting parameters to the original loss functions based on the weight analysis of BPR and triplet loss. Zhuo et al. [62] introduces a flexible boundary instead of the existing fixed boundary. It considers the flexible boundary as the user interest boundary and divides the positive and negative scores based on that boundary.

**Hyperbolic space.** While the previous loss functions all compute in Euclidean space, Yang et al. [52] first introduces the concept of hyperbolic geometry. The geometric properties of hyperbolic space provide rich information for the model training. HRCF [52] computes the distance in hyperbolic space and uses a triplet margin loss. HICF [51] improves the margin learning strategy and negative sampling method in HRCF [52] from the perspective of hyperbolic space.

Graph contrastive learning Lately, Graph Contrastive Learning (GCL)-based models have been demonstrating their effectiveness with high performance [1, 45, 53]. We thus investigate the mathematical differences between the loss functions of GCL-based models and traditional models. Formally, most GCL-based models use a loss function consisting of two terms: recommendation loss and contrastive loss, *i.e.*,  $\mathcal{L}_{GCL} = \mathcal{L}_{Rec} + \lambda \mathcal{L}_{CL}$ . The recommendation loss  $\mathcal{L}_{Rec}$  is fundamentally used in recommender systems. In other words,  $\mathcal{L}_{Rec}$  contains  $\mathcal{L}_{Pointwise}$ ,  $\mathcal{L}_{Pairwise}$ ,  $\mathcal{L}_{Setwise}$ , and  $\mathcal{L}_{AU}$ . In [1, 45, 53], they all use BPR [29] as  $\mathcal{L}_{Rec}$ . By adding the contrastive loss  $\mathcal{L}_{CL}$  to the existing loss, we can expect to see a performance improvement. The contrastive loss  $\mathcal{L}_{CL}$  augments the user and item with two views each and performs contrastive learning on the augmented user and item, *i.e.*,  $\mathcal{L}_{CL} = \mathcal{L}_{CL-user} + \mathcal{L}_{CL-item}$ . GCL-based models are categorized depending on how the augmentation is designed. SGL [45] uses graph augmentation with node/edge dropout, and SimGCL [53] utilizes random noise directly on user/item embeddings as an augmentation. Meanwhile, LightGCL [1] performs SVD on the adjacency matrix and uses the reconstructed graph as an augmentation.

**Others.** In addition to designing a single loss function, research on loss function search [20, 42] and loss function optimization scheduling [48] are also actively studied. Influenced by these studies, recommender systems are also using automatic machine learning to generate loss or customize loss functions [22, 35, 59]. In line with this trend, this paper aims to be a milestone in the development of CF loss functions.