MCL: Mixed-Centric Loss for Collaborative Filtering

Zhaolin Gao*
University of Toronto
Toronto, Canada
zhaolin.gao@mail.utoronto.ca

Zhaoyue Cheng*
Layer6 AI
Toronto, Canada
joey@layer6.ai

Felipe Pérez
Layer 6 AI
Toronto, Canada
felipe@layer6.ai

Jianing Sun
Layer 6 AI
Toronto, Canada
jianing@layer6.ai

Maksims Volkovs
Layer 6 AI
Toronto, Canada
maks@layer6.ai

ABSTRACT

The majority of recent work in latent Collaborative Filtering (CF) has focused on developing new model architectures to learn accurate user and item representations. Typically, a standard pairwise loss function (BPR, Triplet, etc.) is used in these models, and little exploration is done on how to optimally extract signals from the available preference information. In the implicit setting, negative examples are sampled, and these losses allocate weights that solely depend on the difference in user distance between observed (positive) and negative item pairs. This can ignore valuable global information from other users and items, and lead to sub-optimal results. Motivated by this problem, we propose a novel loss which first leverages mining to select the most informative pairs, followed by a weighing process to allocate more weight to harder examples. Our weighting process consists of four different components, and incorporates distance information from other users, enabling the model to better position the learned representations. We conduct extensive experiments and demonstrate that our loss can be applied to different types of CF models leading to significant gains with each type. In particular, by applying our loss to the graph convolutional architecture, we achieve new state-of-the-art results on four different datasets. Further analysis shows that through our loss the model is able to learn better user-item representation space compared to other losses. Full code for this work is available here: https://github.com/layer6ai-labs/MCL.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**; **Personalization**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Recommender Systems, Collaborative Filtering, Personalization, Graph Neural Network

*Authors contributed equally to this work.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04.
https://doi.org/10.1145/3485447.3512106

ACM Reference Format:
Zhaolin Gao, Zhaoyue Cheng, Felipe Pérez, Jianing Sun, and Maksims Volkovs. 2022. MCL: Mixed-Centric Loss for Collaborative Filtering. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512106

1 INTRODUCTION

With the increasing amount of information available online, providing accurate recommendations is becoming essential for online platforms such as multimedia streaming, e-commerce, and social media. Collaborative Filtering (CF) embedding approaches are commonly used to model user and item implicit feedback (e.g. clicks) on these platforms. From matrix factorization [11, 13, 22] to more recent graph-based methods [7, 18, 29, 32, 37], generating high quality user and item embeddings lies at the core of recent progress in recommender systems.

Embedding CF models are commonly trained with a pairwise loss, typically in the form of Bayesian Personalized Ranking (BPR) loss [26] or Triplet loss [9]. Pairwise learning involves pushing positive (interacted) items closer to the user than negative (not interacted) items. To reduce computational complexity negative items are sampled at each iteration. The BPR loss is often used in latent models [6, 14, 23, 26], and more recently in graph convolutional models [7, 12, 18, 39], while Triplet loss is mostly found in deep metric learning approaches [10, 24, 33]. Other commonly used pairwise CF losses are cross entropy [1, 8, 31] and mean squared error [15, 27, 38].

Recent works in recommender systems and other fields such as computer vision, have found that pairwise losses can result in sub-optimal model training due to information loss. In Li et al. [16], the authors argue that the common pairwise losses only use the difference in user distance between positive and negative items, and fail to consider other important information such as relationships between the items themselves. Related works in computer vision [2, 21, 28], show that random sampling of negative items is inefficient at finding hard examples, and can lead to sub-optimal training on easier and less informative data. To address these problems different sampling techniques and training schemes have been proposed in order to fully utilize the information in the preference data [10, 30, 36]. However, this area remains under-explored as research in latent CF has largely focused on designing better model architectures to generate user and item embeddings.

In this work we investigate pairwise learning, and propose a novel learning framework where we combine hard instance mining with a new Mixed-Centric loss function. The loss function allocates weights that depend not only on the relative hardness of each pair but also on the global alignment with other users and items. Specifically, our weighting strategy has four components: a *user-item centric* component that measures how a given item relates to the user, a *same/different-type centric* components that contrast positive and negative items for the user, and a *batch centric* component that provides a global context relating items from different users. We implement our learning framework with three leading CF methods: CML [10], NeuMF [8], and LightGCN [7]. Extensive experiments on multiple public datasets show that training with our approach significantly improves performance of all three models with relative gains of up to 60%. Furthermore, LightGCN trained with our approach achieves new state-of-the-art results on each dataset. Analysis of the learned user/item embeddings reveals that our approach can better separate the embedding space, has stable performance that improves with the number of negative pairs, and is more robust to change in dimension. The major contributions of this paper are summarized below:

- We analyze the traditional losses used for training CF models and evaluate their drawbacks through pairwise weight analysis.
- We propose a loss function that leverages mining to find most informative pairs, and a weighting scheme that combines local information from the target user with global information from other users in the batch.
- We implement our loss on top of existing leading models and demonstrate that it leads to significant improvement in accuracy achieving state-of-the-art results on multiple datasets.
- We analyze the learned embedding space and study the effects of each component including number of negatives, dimension, and other hyper-parameters.

2 RELATED WORK

Pairwise models are common across different fields of machine learning with applications in CF, computer vision, NLP and other areas [7, 28, 36]. A common approach to optimize these models is by randomly sampling pairs throughout learning. However, the candidate space is typically large and can contain many uninformative or easy instances so random sampling can lead to sub-par results. A number of methods have been proposed in CF to address this issue by developing strategies to improve the quality of sampled pairs. Ding et al. [3] proposes to generate high quality pairs by training with reinforcement learning. Park and Chang [25] obtains more informative samples through adversarial sampling and training. Ding et al. [5] shows a relation between score variance and false positives, and uses it to design a sampling strategy. Different from these methods that focuses mostly on generate or get high-quality pairs especially negative samples, we use a simpler mining strategy which works well with the loss and assign different weights to sampled pairs.

In addition to better sampling, other methods have explored different losses to incorporate additional information between users and items. In deep metric learning, Angular Loss [35] takes angle relationship into consideration, while Lifted Structure loss [21] proposes to use pairwise across samples within a batch. In recent CF models, Li et al. [16] adds a component in the loss to push positive and negative item away from each other. Ma et al. [19] incorporates explicit user-user and item-item similarity modeling into the objective function. We build on these approaches and first analyze common pairwise losses from a weight-based perspective. We then propose a new loss that combines negative sample mining with local information from target user and global information from other users in the batch.

3 WEIGHT ANALYSIS FOR PAIRWISE LOSSES

Given a user u, let P_u denote the set of *positive items* that the user has interacted with, and let N_u denote the *negative items* that u hasn't interacted with. A positive pair is defined as $\{u, j\}$ where $j \in P_u$, and a negative pair is defined as $\{u, k\}$ where $k \in N_u$. Latent CF models aim to find embeddings for users and items where proximity between user-item pairs corresponds to relevance. We denote a proximity measure between user u and item i as E_{ui}, common measures include Euclidean distance and dot product. In pairwise learning the loss L is typically designed to push positive items closer to the user and negative ones further apart. The magnitude of the push is determined by the gradient of each user-item pair, and we can analyse it as a weight:

$$w^+_{uj} = \frac{\partial L}{\partial E_{uj}}, \qquad w^-_{uk} = -\frac{\partial L}{\partial E_{uk}} \tag{1}$$

where w^+_{uj} and w^-_{uk} are scalar gradient weights for positive and negative pairs respectively. Using this framework we analyze two commonly used CF losses Triplet and BPR.

Triplet Loss The Triplet loss aims to make the difference between distances for positive and negative pairs to be greater than the margin λ:

$$L_{triplet}(u, j, k) = [E_{uj} - E_{uk} + \lambda]_+ \tag{2}$$

where E is the Euclidean distance. The associated weights are given by:

$$w^+_{uj} = w^-_{uk} = \begin{cases} 1, & \text{if } E_{uj} + \lambda > E_{uk} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The Triplet loss thus selects pairs satisfying $E_{uj} + \lambda > E_{uk}$ and discards those with $E_{uj} + \lambda \leq E_{uk}$. All the selected pairs have the same weight of 1 and discarded pairs have a weight of 0.

BPR Loss The BPR loss aims to make the dot product for a positive pair to be higher than the dot product for a negative pair:

$$L_{BPR}(u, j, k) = \log\ \sigma(E_{uk} - E_{uj}) \tag{4}$$

where E is the dot product between corresponding user and item embeddings, and σ is the sigmoid function. The weights are given by:

$$\begin{aligned} w^+_{uj} = w^-_{uk} &= -\frac{1}{1 + e^{E_{uk} - E_{uj}}} \\ &= -\sigma(E_{uj} - E_{uk}) \end{aligned} \tag{5}$$

BPR thus uses the dot product difference between the positive and negative pairs to allocate the weights. This can be viewed as a soft version of the Triplet loss, instead of using a hinge function with constant weights, BPR weights are scaled with a sigmoid function.

Both Triplet and BPR losses can miss important information during the training process. One one hand, the Triplet loss weighs every qualifying pair equally which can be sub-optimal particularly in the later stages of training where we want the model to focus on the difficult pairs. On the other hand, the BPR loss weights pairs by the dot product difference which allows for dynamic weight allocation to emphasize harder pairs. However, since the pairs are sampled randomly, two pairs with highly varied difficulty could be sampled together, leading to inconsistent weight allocation. For example, if a easy positive pair is sampled with a hard negative pair, the resulting weight can be excessive for the easy positive and insufficient for the hard negative. We also note, that these two losses only rely on the differences between the positive and negative pairs to determine the weights, while ignoring other information in the batch such as relationship to other users and items. Since item embedding space is shared between users, mining information from the batch can provide additional constraints on the embeddings and facilitate learning. In the following section we explore these concepts and propose a new loss to address these shortcomings.

4 OUR APPROACH

In this section we introduce our learning framework that consists of pair mining and our novel Mixed-Centric loss function. Building on the weight analysis in Section 3, we show that under our framework the pair weight consists of four components. Specifically, we show that pair mining can be viewed as *different-type centric* component that contrasts positive and negative items for a given user. Meanwhile, the weighing allocated by the Mixed-Centric loss is directly influenced by three components: *user centric* component measures how a given item relates to the user, *same-type centric* component compares items of the same type (positive or negative) for the user, and *batch centric* component contrasts items across different users. We formally define each component below and discuss their importance.

Pair Mining Influenced by the related work in self-supervised learning from the computer vision domain [36], at each iteration we aim to find hard positive and negative examples for the model to focus on. Conceptually, we define a hard positive example as an item that is further from the user than at least one negative item. Similarly, a hard negative example is an item that is closer to the user than at least one positive item. This forms the basis of our mining procedure where at each iteration we discard items that don't pass the hardness criteria. To account for the fact that the embedding space (and corresponding distances) is continuously changing throughout learning, we add a margin when comparing distances between items. Formally, a positive pair $\{u, j\}$ is selected if:

$$E_{uj} > \min_{k \in N_u} E_{uk} - \epsilon \tag{6}$$

where E is the Euclidean distance, N_u is the set of negative items for u, and ϵ is a margin parameter that controls the degree of separation. Similarly, a negative pair $\{u, k\}$ is selected if:

$$E_{uk} < \max_{j \in P_u} E_{uj} + \epsilon \tag{7}$$

where P_u is the set of positive items for u. The mining procedure is illustrated in Figure 1, and can be viewed as a form of binary hinge weight where selected items have a weight of 1 and discarded items have a weight of 0. In practice the set of negative items N_u can be very large, to reduce computational complexity at each iteration we randomly sample a small subset of negative items and apply the mining procedure to that subset.

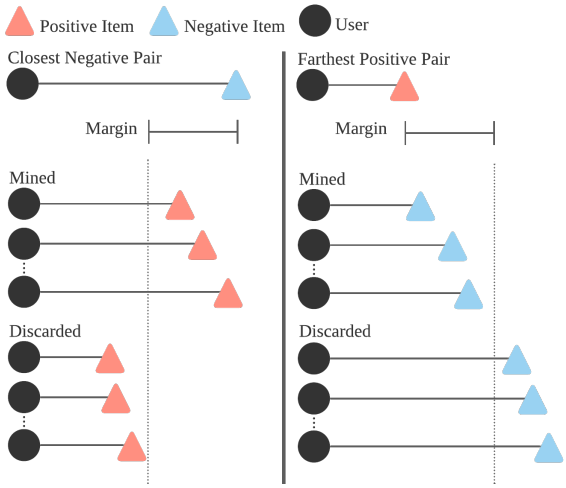


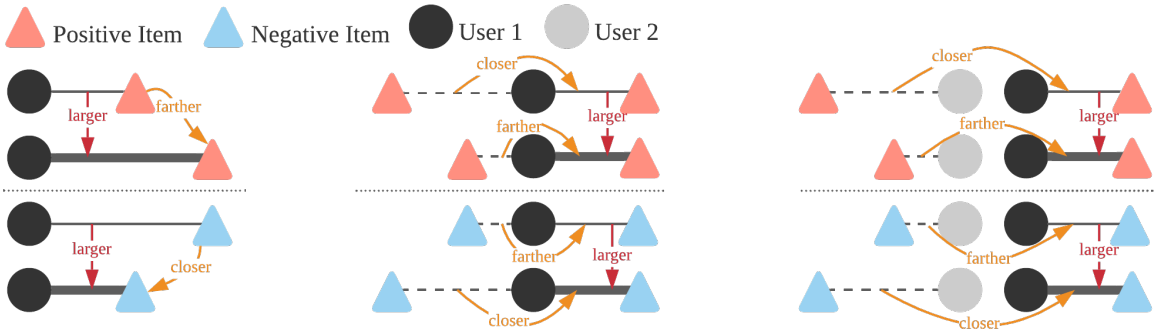
Figure 1: Illustration of the pair mining process, line length represents distance to the user. Left: positive items whose distance to the user is smaller than the distance of the closest negative item minus the margin are discarded, all others are kept. Right: negative items whose distance to the user is farther than the distance of the farthest positive item plus the margin are discarded, all others are kept.

Mixed-Centric Loss (MCL) After pair mining, we denote the selected positive and negative items for user u as P_u^s and N_u^s respectively. Given a batch of m users B, we define our loss as:

$$\begin{aligned} L_{MCL} = &\frac{1}{\alpha} \log[1 + \frac{1}{m} \sum_{u \in B} \sum_{j \in P_u^s} e^{\alpha(E_{uj} + \lambda_p)}] \\ &+ \frac{1}{\beta} \log[1 + \frac{1}{m} \sum_{u \in B} \sum_{k \in N_u^s} e^{-\beta(E_{uk} + \lambda_n)}] \end{aligned} \tag{8}$$

Here, λ_p and λ_n are hyper-parameters controlling the margin allowance for the positive and negative pairs, and α, β control the loss contribution from positive and negative pairs respectively. The first term in Equation 8 aims to lower the distance for all selected positive pairs, and the second term aims to raise it for all selected negative pairs. Under this loss the weight for a positive pair $\{u, j\}$ is given by:

$$\begin{aligned} w_{uj}^+ &= \frac{1}{m} \cdot \frac{e^{\alpha E_{uj}}}{e^{-\alpha\lambda_p} + \frac{1}{m} \sum_{u' \in B} \sum_{i \in P_{u'}^s} e^{\alpha E_{u'i}}} \\ &= \frac{1}{m} \cdot \frac{1}{w_1^+(u,j) + w_2^+(u,j) + w_3^+(u,j)} \end{aligned} \tag{9}$$



(a) User-Item Centric (b) Same-type Centric (c) Batch Centric

Figure 2: Illustration of the three weighting components in our Mixed-Centric loss. Circles represent users and triangles represent items. Length of each line represents distance between the corresponding user-item pair. The width of the line represents the weight on that pair – the thicker the line, the larger the weight.

This weight can be partitioned into three components w_1^+, w_2^+ and w_3^+:

$$w_1^+(u,j) = e^{-\alpha(E_{uj}+\lambda_p)}$$

$$w_2^+(u,j) = \frac{1}{m}\sum_{i\in P_u^s} e^{\alpha(E_{ui}-E_{uj})}$$

$$w_3^+(u,j) = \frac{1}{m}\sum_{\substack{u'\in B\\ u'\neq u}}\sum_{i\in P_{u'}^s} e^{\alpha(E_{u'i}-E_{uj})}$$

Similarly, the weight for a negative pair $\{u,k\}$ can be decomposed into three analogous components:

$$w_{uk}^- = \frac{1}{m}\cdot\frac{1}{w_1^-(u,k)+w_2^-(u,k)+w_3^-(u,k)} \tag{10}$$

where:

$$w_1^-(u,k) = e^{\beta(E_{uk}+\lambda_n)}$$

$$w_2^-(u,k) = \frac{1}{m}\sum_{i\in N_u^s} e^{\beta(E_{uk}-E_{ui})}$$

$$w_3^-(u,k) = \frac{1}{m}\sum_{\substack{u'\in B\\ u'\neq u}}^{m}\sum_{i\in N_{u'}^s} e^{\beta(E_{uk}-E_{u'i})}$$

For brevity, we focus on illustrating how the positive components w_1^+, w_2^+, and w_3^+ influence the overall weight w^+, as analogous arguments hold for the negative weights. w_1^+ is the user-item centric component and depends only on the distance between u and j. Harder positive items that are further from the user have a smaller w_1^+ and larger overall weight w_{uj}^+, this effect is illustrated in Figure 2(a). In w_2^+, the target item j is compared with all other positive items in P_u^s that pass the mining criteria. This component can thus be viewed as same-type centric, where if distance E_{uj} is larger than distances between u and other items of the same type, the overall weight w_{uj}^+ is increased. This has a regularization effect on the item embedding space where the loss favours to have all items of the same type within a similar distance to the user, and outliers are strongly penalised; Figure 2(b) demonstrates this effect. Finally, w_3^+ compares target pair distance E_{ui} with other user-item distances $E_{u'i}$ within the batch, and can be interpreted as the batch-centric component. This component provides additional consistency across users, where the loss aims to place all positive items within the same distance for each user. Pairs that are significantly further away have a larger overall weight, and are emphasized during training, this is illustrated in Figure 2(c).

Jointly, these components convey both local context on how the target item relates to the user, and global context on how the item relates to other items of the same type from the target user as well as other users in the dataset. Combining local and global contexts enables the model to enforce stricter consistency on the embedding space which, as we show in experiments section, leads to better separation of the user-item embeddings and significant accuracy improvement.

Table 1: Complexity analysis. N denotes the total number of pairs, U denotes the total number of the users, I denotes the total number of the items, D denotes the embedding dimension, and S denotes the memory size in SRNS [4].

Method	Time Complexity	Space Complexity
BPR [26]	$O(ND)$	$O((U+I)D)$
Triplet [9]	$O(ND)$	$O((U+I)D)$
SML [16]	$O(ND)$	$O((U+I)D)$
SRNS [4]	$O(NDS)$	$O((U+I)D)$
MCL	$O(ND)$	$O((U+I)D)$

Table 2: Dataset statistics.

Dataset	#User	#Item	#Interactions	Density
Amazon-Digital-Music	5,541	3,568	46,846	0.237%
Amazon-Grocery	14,684	8,713	108,017	0.084%
Amazon-Books	52,406	41,264	1,856,747	0.086%
Yelp2021	97,462	48,294	2,209,755	0.047%

Complexity Analysis

The time complexity of MCL can be split into two parts: pair mining and loss computation. Given a user u, let $n_u \ll |N_u|$ denote the number of negative pairs sampled for this user, and $p_u = |P_u|$ the number of positive pairs. The total number of sampled pairs across all users is denoted by N where $N = \sum_{u\in U} p_u + n_u$. We note that the

Table 3: Recall (top) and NDCG (bottom) results for all datasets with CML, NeuMF, and LightGCN as base models. The best-performing model for each dataset and base model is highlighted in bold and next best model is underlined. Models trained with SML, SRNS, and our loss are indicated with "+SML", "+SRNS", and "+MCL" respectively. Relative improvements comparing to the second best model are shown in brackets. Asterisks denote statistically significant improvements according to the Wilcoxon signed-rank test.

Datasets		CML	CML +SML	CML +SRNS	CML +MCL	NeuMF	NeuMF +SML	NeuMF +SRNS	NeuMF +MCL	LightGCN	LightGCN +SML	LightGCN +SRNS	LightGCN +MCL
Amazon-Digital-Music	R@5	0.0405	0.0453	0.0415	**0.0459** (+1.32%)	0.0875	0.1203	0.1137	**0.1429** (+18.8%*)	0.1426	0.1289	0.1518	**0.1612** (+6.19%*)
	R@10	0.1157	0.1215	0.1253	**0.1400** (+11.7%*)	0.1330	0.1825	0.1615	**0.2120** (+16.2%*)	0.2100	0.1916	0.2078	**0.2308** (+9.90%*)
	R@20	0.2129	0.2169	0.2234	**0.2447** (+9.53%*)	0.1976	0.2520	0.2206	**0.2937** (+16.5%*)	0.2802	0.2652	0.2821	**0.3082** (+9.25%*)
Amazon-Grocery	R@5	0.0303	0.0318	0.0325	**0.0365** (+12.3%*)	0.0367	0.0436	0.0407	**0.0585** (+34.2%*)	0.0590	0.0473	0.0607	**0.0699** (+15.2%*)
	R@10	0.0643	0.0683	0.0671	**0.0732** (+7.17%*)	0.0595	0.0659	0.0642	**0.0896** (+36.0%*)	0.0888	0.0780	0.0909	**0.1051** (+15.6%*)
	R@20	0.1152	0.1185	0.1165	**0.1230** (+3.80%)	0.0951	0.1024	0.0946	**0.1360** (+32.8%*)	0.1306	0.1204	0.1353	**0.1492** (+10.3%*)
Amazon-Books	R@5	0.0417	0.0510	0.0531	**0.0608** (+14.5%*)	0.0296	0.0346	0.0234	**0.0488** (+41.0%*)	0.0542	0.0476	0.0465	**0.0634** (+17.0%*)
	R@10	0.0695	0.0832	0.0853	**0.0957** (+12.2%*)	0.0494	0.0583	0.0402	**0.0802** (+37.6%*)	0.0871	0.0774	0.0762	**0.1000** (+14.8%*)
	R@20	0.1102	0.1300	0.1330	**0.1450** (+9.02%*)	0.0804	0.0944	0.0659	**0.1253** (+32.7%*)	0.1348	0.1193	0.1181	**0.1495** (+10.9%*)
Yelp2021	R@5	0.0234	0.0300	0.0267	**0.0316** (+5.33%)	0.0202	0.0202	0.0162	**0.0298** (+47.5%*)	0.0327	0.0242	0.0296	**0.0361** (+10.4%*)
	R@10	0.0421	0.0522	0.0483	**0.0555** (+6.32%*)	0.0348	0.0356	0.0282	**0.0514** (+44.4%*)	0.0542	0.0423	0.0502	**0.0607** (+12.0%*)
	R@20	0.0738	0.0877	0.0773	**0.0921** (+5.02%)	0.0581	0.0595	0.0471	**0.0861** (+44.7%*)	0.0881	0.0708	0.0823	**0.0976** (+10.8%*)

Datasets		CML	CML +SML	CML +SRNS	CML +MCL	NeuMF	NeuMF +SML	NeuMF +SRNS	NeuMF +MCL	LightGCN	LightGCN +SML	LightGCN +SRNS	LightGCN +MCL
Amazon-Digital-Music	N@5	0.0209	0.0218	0.0213	**0.0223** (+2.29%)	0.0721	0.0896	0.0863	**0.1093** (+22.0%*)	0.1076	0.0973	0.1138	**0.1211** (+6.41%*)
	N@10	0.0471	0.0501	0.0513	**0.0550** (+7.21%*)	0.0845	0.1110	0.1024	**0.1334** (+20.2%*)	0.1312	0.1195	0.1337	**0.1453** (+8.68%*)
	N@20	0.0743	0.0766	0.0786	**0.0841** (+7.00%*)	0.1022	0.1311	0.1192	**0.1563** (+19.2%*)	0.1514	0.1405	0.1550	**0.1676** (+8.13%*)
Amazon-Grocery	N@5	0.0171	0.0203	0.0204	**0.0232** (+13.7%*)	0.0274	0.0319	0.0294	**0.0415** (+30.1%*)	0.0441	0.0340	0.0439	**0.0507** (+15.0%*)
	N@10	0.0290	0.0327	0.0323	**0.0342** (+4.59%)	0.0354	0.0398	0.0378	**0.0524** (+31.7%*)	0.0545	0.0448	0.0546	**0.0631** (+15.6%*)
	N@20	0.0432	0.0468	0.0451	**0.0485** (+3.63%)	0.0455	0.0501	0.0456	**0.0657** (+31.1%*)	0.0664	0.0568	0.0672	**0.0757** (+12.6%*)
Amazon-Books	N@5	0.0606	0.0732	0.0756	**0.0881** (+16.5%*)	0.0443	0.0510	0.0343	**0.0711** (+39.4%*)	0.0771	0.0686	0.0668	**0.0927** (+20.2%*)
	N@10	0.0681	0.0817	0.0837	**0.0961** (+14.8%*)	0.0492	0.0574	0.0390	**0.0792** (+38.0%*)	0.0854	0.0764	0.0744	**0.1010** (+18.3%*)
	N@20	0.0824	0.0979	0.1021	**0.1127** (+10.4%*)	0.0601	0.0702	0.0483	**0.0949** (+35.2%*)	0.1020	0.0911	0.0893	**0.1175** (+15.2%*)
Yelp2021	N@5	0.0277	0.0356	0.0298	**0.0376** (+5.62%)	0.0241	0.0240	0.0191	**0.0355** (+47.3%*)	0.0387	0.0287	0.0354	**0.0437** (+12.9%*)
	N@10	0.0342	0.0430	0.0371	**0.0457** (+6.28%*)	0.0290	0.0294	0.0232	**0.0427** (+45.2%*)	0.0457	0.0350	0.0422	**0.0516** (+12.9%*)
	N@20	0.0451	0.0552	0.0481	**0.0582** (+5.43%)	0.0370	0.0378	0.0298	**0.0546** (+44.4%*)	0.0574	0.0450	0.0532	**0.0641** (+11.7%*)

complexity of computing the score for pair pair has the same order as the embedding dimension D, and so we have that computing the scores for all relevant pairs has complexity $O(ND)$. Once these scores are obtained, Equations 6 and 7 add an additional cost of $n_u + p_u$ per user. The rest of the computation is spent on selecting the positive and negative pairs via these equations which adds N comparisons between the pair score and the max/min. Therefore, the total complexity of pair mining is:

$$O(ND + \sum_{u \in U} (n_u + p_u) + N) = O(N(D + 2)) \tag{11}$$

The loss computation for each selected pair is constant, and therefore the extra amount of computations is bounded by N. We get that the total complexity is:

$$O(N(D + 2) + N) = O(N(D + 3)) = O(ND) \tag{12}$$

The time complexity of MCL is the same as BPR, Triplet (see Table 1), and SML while SRNS has higher complexity due to score computation in the memory bank.

For space complexity, SML is the only method that requires additional space to store user and item biases, while the other four methods only need the space for the embeddings. However, the addition in space is constant for SML so the space complexity for all five methods is the same. The complexity analysis shows that our loss has the same complexity in time and space as BPR, Triplet, and SML while SRNS has higher time complexity.

5 EXPERIMENTS

We evaluate our method on public datasets and compare against leading CF baselines. We use the Amazon-Digital-Music, Amazon-Grocery, Amazon-Books, [20] and Yelp2021 datasets. The statistics for these datasets are summarized in Table 2. The datasets vary in size and density, providing a comprehensive view of model performance. Following previous work [16, 18, 40], we randomly select 80% of the interactions for training and 10% for validation and 10% for testing. To evaluate top-k ranking performance for each model, we adopt two widely used metrics: Recall and Normalized Discounted Cumulative Gain (NDCG) [18, 34, 37].

Baselines We use CML [10], NeuMF [8], and LightGCN [7] as base models since they represent the leading models in metric learning, neural network and graph convolutional CF literature. We apply state-of-the-art loss functions and our proposed MCL approach to the base models to measure the relative improvement. For the loss functions we leverage:

- **SML** [16] introduces an adaptive bias for each user to allow different base preferences, and a symmetric item-centric metric to push away negative items from positive while maintaining close distance between user and positive item.
- **SRNS** [4] reduces the false negative instances during the sampling process by favoring high-variance negative items in the memory.

In addition to the loss functions, we also compare our results with other leading CF models:

Table 4: Recall (top) and NDCG (bottom) results for all datasets with VAE-CF, BPRMF, LRML, NGCF, and IMP-GCN as baselines. The best-performing model for each dataset and metric is highlighted in bold and next best model is underlined. Models trained with our loss are indicated with "+MCL" and relative improvements are shown in brackets. Asterisks denote statistically significant improvements according to the Wilcoxon signed-rank test.

Datasets		VAE-CF	BPRMF	LRML	NGCF	IMP-GCN	LightGCN+MCL
Amazon-Digital-Music	R@5	0.1211	0.1181	0.0496	0.1144	0.1493	**0.1612** (+7.97%*)
	R@10	0.1730	0.1751	0.0766	0.1689	0.2052	**0.2308** (+12.5%*)
	R@20	0.2469	0.2431	0.1163	0.2450	0.2773	**0.3082** (+11.1%*)
Amazon-Grocery	R@5	0.0483	0.0451	0.0212	0.0320	0.0602	**0.0699** (+16.1%*)
	R@10	0.0774	0.0743	0.0357	0.0576	0.0897	**0.1051** (+17.2%*)
	R@20	0.1166	0.1075	0.0573	0.0928	0.1278	**0.1492** (+16.7%*)
Amazon-Books	R@5	0.0371	0.0349	0.0344	0.0452	0.0565	**0.0634** (+12.2%*)
	R@10	0.0564	0.0589	0.0561	0.0728	0.0911	**0.1000** (+9.77%*)
	R@20	0.0825	0.0941	0.0868	0.1135	0.1413	**0.1495** (+5.80%)
Yelp2021	R@5	0.0204	0.0241	0.0185	0.0280	0.0332	**0.0361** (+8.73%*)
	R@10	0.0331	0.0423	0.0317	0.0479	0.0568	**0.0607** (+6.87%*)
	R@20	0.0528	0.0703	0.0523	0.0785	0.0935	**0.0976** (+4.39%)

Datasets		VAE-CF	BPRMF	LRML	NGCF	IMP-GCN	LightGCN+MCL
Amazon-Digital-Music	N@5	0.0912	0.0919	0.0385	0.0851	0.1096	**0.1209** (+10.3%*)
	N@10	0.1105	0.1120	0.0479	0.1044	0.1296	**0.1468** (+13.3%*)
	N@20	0.1317	0.1316	0.0593	0.1259	0.1502	**0.1684** (+12.1%*)
Amazon-Grocery	N@5	0.0364	0.0331	0.0153	0.0232	0.0445	**0.0507** (+13.9%*)
	N@10	0.0464	0.0433	0.0206	0.0323	0.0549	**0.0631** (+14.9%*)
	N@20	0.0576	0.0529	0.0268	0.0421	0.0657	**0.0757** (+15.2%*)
Amazon-Books	N@5	0.0619	0.0388	0.0525	0.0654	0.0811	**0.0927** (+14.3%*)
	N@10	0.0632	0.0524	0.0575	0.0722	0.0897	**0.1010** (+12.6%*)
	N@20	0.0711	0.0683	0.0679	0.0864	0.1093	**0.1175** (+7.50%*)
Yelp2021	N@5	0.0260	0.0234	0.0237	0.0342	0.0396	**0.0437** (+10.4%*)
	N@10	0.0296	0.0326	0.0274	0.0407	0.0474	**0.0516** (+8.86%*)
	N@20	0.0362	0.0437	0.0344	0.0511	0.0594	**0.0641** (+7.91%*)

- **VAE-CF** [17] is a Bayesian collaborative filtering approach based on variational autoencoders.
- **BPR** [26] proposes to use BPR loss with matrix factorization (MF).
- **LRML** [33] employs an augmented memory module to construct latent relationships instead of simple push-pull mechanisms for deep metric learning.
- **NGCF** [37] applies graph convolutional networks to user-item bipartile graph by performing embedding propagation.
- **IMP-GCN** [18] divides the user-item graph into subgraphs and uses higher-order graph convolution inside subgraphs.

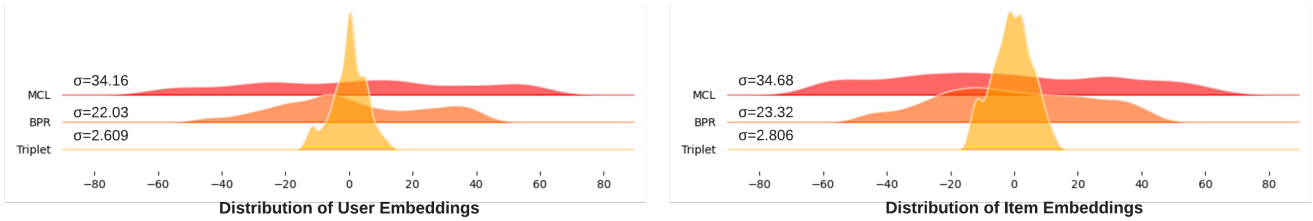
Implementation Details For all models, we set the embedding dimension to 64 and number of negative pairs for each positive pair to 10 to make the comparison fair. We set other hyper-parameters for each baseline using cross validation around the best settings suggested by the respective authors. For temperature parameters, we select α from {1, $\frac{5}{4}$, $\frac{5}{3}$, $\frac{5}{2}$, 5} and β from {1, 2, 3, 4, 5}. The margin parameters, λ_p and λ_n are chosen from {0, 0.5, 1,..., 9.5, 10} and {-3, -2.5,..., 2.5, 3} respectively. The mining margin ϵ is fixed to 1. All hyper-parameters for the Mixed-Centric loss are set through cross validation and we discuss their effects in the ablation analysis. We set the batch size to 1K users for each dataset and use the Adam optimizer, all other hyper-parameters for CML, NeuMF, and LightGCN are set to defaults. Experiments are conducted with PyTorch on a server with 40 Intel Xeon CPU@2.20GHz cores and Nvidia Titan V GPU.

5.1 Performance Comparison With Other Losses

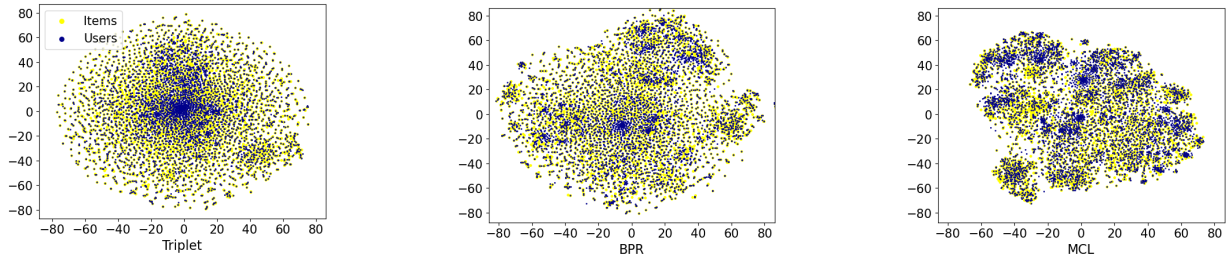
The results for the four datasets with different loss functions are shown in Table 3. We denote models trained with SML, SRNS, and our loss as " +SML", " +SRNS", and " +MCL" respectively with CML, NeuMF, and LightGCN as the base models. We can see that adding MCL significantly improves performance for all models on each of the four datasets and models with MCL achieves highest performance compared to the same model with SML and SRNS, demonstrating the effectiveness of MCL loss when combined with different base models. Among the baselines, we found that when using NeuMF as the base model, SML consistently performs better than SRNS on all datasets while SRNS performs better when using CML and LightGCN as base models.

5.2 Performance Comparison With Other Methods

Table 4 shows the performance comparison results with other state-of-the-art collaborative filtering approaches. Among the baseline models, the performance of VAE-CF is on par with NGCF and both perform better than LRML and BPRMF. IMP-GCN has the



(a) User and item embeddings projected to 1D, the standard deviation (σ) is shown on the left.



(b) User and item embeddings projected to 2D.

Figure 3: Visualization of the learned embeddings for three losses. The visualizations are obtained by training the same LightGCN model architecture on the Amazon-Digital-Music dataset and then projecting the learned embeddings to 1D (Figure 3a) and 2D (Figure 3b) spaces with t-SNE. Recall@20 for the three losses are as follows Triplet:0.2656**, BPR:**0.2802**, and MCL:**0.3082**.**

strongest performance amongst the baselines on all four datasets. Our proposed loss added to the LightGCN model (LightGCN + MCL) achieves new state-of-the-art results on all datasets and consistently outperforms all other approaches. In particular, when compared to the strongest baseline IMP-GCN in terms of NDCG@20, our model reaches relative improvements of 12.1%, 15.2%, 7.5%, and 7.91% on the Amazon-Digital-Music, Amazon-Grocery, Amazon-Books and Yelp2021 datasets respectively. These results demonstrate that by leveraging information more effectively in the preference data and placing additional constraints on the embedding space, we can achieve significant gains in performance without changing the underlying model architecture.

5.3 Embedding Visualization

An important factor for the high performance of our loss is the *batch centric* component that adds global information from other users within the batch. We discussed that this component acts as a regularizer and encourages items of the same type (positive or negative) to be within comparable distance for every user. To further evaluate the effect that MCL learning has on the embedding space we visualize the learned embeddings by projecting them to one-dimensional (1D) and two-dimensional (2D) spaces with t-SNE. We use the same LightGCN model architecture and train it with the Triplet, BPR, and MCL losses on the Amazon-Digital-Music dataset, plots of the projected embeddings are shown in Figure 3. We recognize that aggressive dimensionality reduction can skew the embedding space, and have empirically verified that the projected representations approximate the user-item distances reasonably well.

Figure 3a shows the distributions of the 1D projected user and item embeddings and the corresponding standard deviations. Both user and item distributions for the Triplet loss are highly peaked with a small standard deviation. This effect can be further observed in the 2D projection plot in Figure 3b. Triplet loss projection has a radial shape with the majority of users in the middle, and items grouped in a circular region around users. Both plots exemplify one of the major drawbacks of the Triplet loss: the weight is not distributed based on the difficulty of each pair. So easy positives get a lot of weight, pulling users and positive items together to the center and forming large clusters. Since users that are in close proximity get similar recommendations, large user clusters can have undesirable effects where it is difficult for the model to identify specific users and provide truly personalized recommendations.

The BPR loss is able to correct some of these drawbacks by using a dynamic weighting scheme that depends on the relative difference between pair distances. We see that BPR's 1D projection has more than 10x larger standard deviation than the Triplet loss, and the 2D projection shows clear user sub-clusters that are spread throughout the embedding space. However, there is still a large cluster of users and items in the center. That cluster contains over 30% of all users within close proximity to each other and can lead to sub-optimal recommendations. Our MCL loss further increases the standard deviation by over 50% relative to BPR, and from the 2D projection we see that both users and items are now separated into much tighter clusters that span the whole range of the embedding space. This in turn leads to a significant accuracy improvement where MCL boosts Recall@20 by 16% and 10% over Triplet and BPR losses respectively.

5.4 Ablation Analysis

Number of Negatives Number of negative samples plays an important role in MCL particularly for the pair mining procedure. Distance to the furthest positive item can be computed exactly for each user, but distance to the closest negative item is estimated from the negative samples. Consequently, larger sample set leads to a better distance approximation which in turn provides a better

Table 5: Recall@20 on Amazon-Digital-Music and Amazon-Books datasets as the number of negatives (n) is varied from 1 to 20.

Dataset	Model	n=1	n=2	n=3	n=5	n=10	n=20
Amazon-Digital-Music	LightGCN	**0.2880**	0.2831	0.2820	0.2812	0.2801	0.2771
	LightGCN+MCL	0.2962	0.3038	0.3056	0.3071	0.3082	**0.3093**
Amazon-Books	LightGCN	**0.1411**	0.1386	0.1375	0.1362	0.1348	0.1335
	LightGCN+MCL	0.1233	0.1365	0.1425	0.1451	0.1495	**0.1532**

Table 6: Recall@20 on Amazon-Digital-Music and Amazon-Books datasets as embedding dimension d is varied from 16 to 64.

Model	Amazon-Digital-Music			Amazon-Books		
	d=64	d=32	d=16	d=64	d=32	d=16
LightGCN	0.2801	0.2701	0.2407	0.1348	0.1088	0.0841
LightGCN+MCL	0.3082	0.2991	0.2577	0.1495	0.1256	0.097
Rel. Improv.	**10.03%**	**10.74%**	**7.06%**	**10.91%**	**15.44%**	**15.34%**

estimate of the hard positive items that the model should focus on (see Figure 1). However, the computation in the pair mining procedure scales linearly with the number of samples so there is a direct trade-off between estimate accuracy and computational complexity. To evaluate the effect of the negative sample set size, we conduct an ablation study and vary the number of negatives from 1 to 20. We compare performance between LightGCN and LightGCN+MCL where both models get the same set of negative samples at each iteration, results are shown in Table 5. We see that LightGCN+MCL steadily improves performance with larger sample sizes while LightGCN gradually degrades. LightGCN uses the BPR loss so with more negative samples the weight on each positive pair increases. This can amplify the clustering effect and pull positive items closer towards the users making the embedding distribution narrower. We have empirically verified this with the t-SNE analysis. In MCL, regularization from same-type and batch centric components prevent the model from over fitting on easy positive/negative items, and LightGCN+MCL benefits from more accurate mining and better estimate of the batch centric component. Even with only a few negatives such as $n = 3$, LightGCN+MCL is already able to outperform the LightGCN counterpart.

Embedding Dimension Embedding dimension is one of the most important parameters in latent CF since it directly controls the representational power of the model. To assess the effect of this parameter, we ablate embedding dimension by varying it from 16 to 64. We again use the strongest LightGCN baseline and compare it with our LightGCN+MCL version. The results are shown in Table 6, and we see that LightGCN+MCL outperforms LightGCN on all settings but the performance for both models drops as dimensionality is reduced. We can potentially attribute this improvement to the more spread out embedding distribution under MCL (as shown in Figure 3b). This allows the model to take a better advantage of the embedding space and counteract the reduction in dimensionality. Notably, LightGCN+MCL at $d = 32$ on Amazon-Digital-Music has higher performance than LightGCN at $d = 64$, demonstrating the significance of improvement.

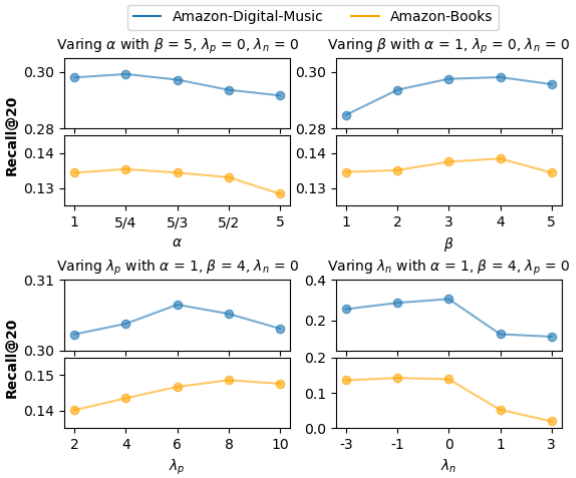


Figure 4: LightGCN+MCL recall@20 hyper-parameter search results for Amazon-Digital-Music and Amazon-Books datasets.

Other Hyper-Parameters To evaluate the effect of other hyper-parameters on our loss, we test different settings with $\beta = 5$, $\alpha \in \{1, \frac{5}{4}, \frac{5}{3}, \frac{5}{2}, 5\}$ and $\alpha = 1$, $\beta \in \{1, 2, 3, 4, 5\}$ while λ_p and λ_n are fixed to 0. Then, we test the $\lambda_p \in \{2, 4, 6, 8, 10\}$ and $\lambda_n \in \{-3, -1, 0, 1, 3\}$ while fixing α and β to the best settings from the previous test. The results are shown in Figure 4 for the LightGCN+MCL model trained on Amazon-Digital-Music and Amazon-Books datasets. We have two main observations:

- For both datasets, the peak performance occurs when the ratio α/β is 4 ($\alpha = \frac{5}{4}$, $\beta = 5$ for Amazon-Digital-Music; $\alpha = 1$, $\beta = 4$ for Amazon-Books). Since the ratio of α and β determines the weight split between positive and negative pairs, the result indicates that the positive pairs should receive four times more weights than the negative pairs to yield the best result.
- For both datasets, the optimal value for λ_p lies in [6, 8] while for λ_n is in [-1, 0]. This difference is due to the Euclidean distance separation between positive and negative pairs. Since $E_{uj} \ll E_{uk}$, we need a large λ_p and a small λ_n to make the magnitudes of $E_{uj}+\lambda_p$ and $E_{uk}+\lambda_n$ similar in the user-item centric components $w_1(u, j)$ and $w_1(u, k)$ (see Equations 9 and 10). Increasing λ_n hurts the model significantly as $w_1(u, k)$ grows exponentially.

6 CONCLUSION

We present a new learning framework for implicit CF that consists of sample mining and MCL loss. We additionally propose a weight-based analysis of pairwise losses and show that weighting in our loss consists of four components that richly describe the preference information available in each batch. We implement MCL with three leading CF model and show significant improvement with each model. In particular, when MCL is combined with the LightGCN architecture we archive new state-of-the-art on all datasets. Analysis of learned embedding space demonstrates that MCL achieves better user-item separation and is robust to different hyper-parameters.

REFERENCES

[1] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[2] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1. IEEE, 539–546.

[3] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data.. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[4] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. [n. d.]. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. In *Advances in Neural Information Processing Systems*.

[5] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. *arXiv preprint arXiv:2009.03376* (2020).

[6] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[7] 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 ACM SIGIR International Conference on Research and Development in Information Retrieval*.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the International World Wide Web Conference*.

[9] E Hoffer and N Ailon. 2014. Deep metric learning using Triplet network. arXiv. *arXiv preprint arXiv:1412.6622* (2014).

[10] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the International World Wide Web Conference*.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining*.

[12] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*.

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

[14] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using Bayesian personalized ranking for social network data. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[15] Oleksii Kuchaiev and Boris Ginsburg. 2017. Training deep autoencoders for collaborative filtering. *arXiv preprint arXiv:1708.01715* (2017).

[16] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*.

[17] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the International World Wide Web Conference*.

[18] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware Message-Passing GCN for Recommendation. In *Proceedings of the International World Wide Web Conference*.

[19] 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 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[20] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[21] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[22] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the IEEE International Conference on Data Mining*.

[23] Weike Pan and Li Chen. 2013. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[24] Chanyoung Park, Donghyun Kim, Xing Xie, and Hwanjo Yu. 2018. Collaborative translational metric learning. In *Proceedings of the IEEE International Conference on Data Mining*.

[25] Dae Hoon Park and Yi Chang. 2019. Adversarial sampling and training for semi-supervised information retrieval. In *Proceedings of the International World Wide Web Conference*.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[27] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the International World Wide Web Conference*.

[28] Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*.

[29] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. 2021. HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering. In *Proceedings of the Web Conference 2021*.

[30] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. 2020. Circle Loss: A Unified Perspective of Pair Similarity Optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[31] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[32] Zhulin Tao, Yinwei Wei, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. 2020. MGAT: multimodal graph attention network for recommendation. *Information Processing & Management* 57, 5 (2020), 102277.

[33] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the International World Wide Web Conference*.

[34] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. 2018. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*.

[35] 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*.

[36] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. 2019. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the ACM SIGIR International conference on Research and development in Information Retrieval*.

[38] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[39] Yu Zheng, Chen Gao, Xiangnan He, Yong Li, and Depeng Jin. 2020. Price-aware recommendation with graph convolutional networks. In *Proceedings of the International Conference on Data Engineering*.

[40] Jin Peng Zhou, Zhaoyue Cheng, Felipe Pérez, and Maksims Volkovs. 2020. TAFA: Two-headed attention fused autoencoder for context-aware recommendations. In *Proceedings of the International World Wide Web Conference*.