On the Effectiveness of Sampled Softmax Loss for Item Recommendation

JIANCAN WU, University of Science and Technology of China, China
XIANG WANG, University of Science and Technology of China, China
XINGYU GAO, Institute of Microelectronics of the Chinese Academy of Sciences, China
JIAWEI CHEN, Zhejiang University, China
HONGCHENG FU, Tencent Music Entertainment Group, China
TIANYU QIU, Tencent Music Entertainment Group, China
XIANGNAN HE, University of Science and Technology of China, China

The learning objective plays a fundamental role to build a recommender system. Most methods routinely adopt either pointwise (*e.g.,* binary cross-entropy) or pairwise (*e.g.,* BPR) loss to train the model parameters, while rarely pay attention to softmax loss, which assumes the probabilities of all classes sum up to 1, due to its computational complexity when scaling up to large datasets or intractability for streaming data where the complete item space is not always available. The sampled softmax (SSM) loss emerges as an efficient substitute for softmax loss. Its special case, InfoNCE loss, has been widely used in self-supervised learning and exhibited remarkable performance for contrastive learning. Nonetheless, limited recommendation work uses the SSM loss as the learning objective. Worse still, none of them explores its properties thoroughly and answers "Does SSM loss suit for item recommendation?" and "What are the conceptual advantages of SSM loss, as compared with the prevalent losses?", to the best of our knowledge.

In this work, we aim to offer a better understanding of SSM for item recommendation. Specifically, we first theoretically reveal three model-agnostic advantages: (1) mitigating popularity bias, which is beneficial to long-tail recommendation; (2) mining hard negative samples, which offers informative gradients to optimize model parameters; and (3) maximizing the ranking metric, which facilitates top-K performance. However, based on our empirical studies, we recognize that the default choice of cosine similarity function in SSM limits its ability in learning the magnitudes of representation vectors. As such, the combinations of SSM with the models that also fall short in adjusting magnitudes (*e.g.,* matrix factorization) may result in poor representations. One step further, we provide mathematical proof that message passing schemes in graph convolution networks can adjust representation magnitude according to node degree, which naturally compensates for the shortcoming of SSM. Extensive experiments on four benchmark datasets justify our analyses, demonstrating the superiority of SSM for item recommendation. Our implementations are available in both TensorFlow[1] and PyTorch[2].

[1]https://github.com/wujcan/SSM-TensorFlow
[2]https://github.com/wujcan/SSM-Torch

Authors' addresses: Jiancan Wu, wujcan@gmail.com, University of Science and Technology of China, 443 Huangshan Road, Hefei, China, 230027; Xiang Wang, xiangwang1223@gmail.com, University of Science and Technology of China, 443 Huangshan Road, Hefei, China, 230027; Xingyu Gao, gxy9910@gmail.com, Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China, 100029; Jiawei Chen, sleepyhunt@zju.edu.cn, Zhejiang University, Hangzhou, China, 310058; Hongcheng Fu, darrenfu@tencent.com, Tencent Music Entertainment Group, Shenzhen, China, 518000; Tianyu Qiu, timmyqiu@tencent.com, Tencent Music Entertainment Group, Shenzhen, China, 518000; Xiangnan He, xiangnanhe@gmail.com, University of Science and Technology of China, 443 Huangshan Road, Hefei, China, 230027.

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 ACM 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.
© 2018 Association for Computing Machinery.
0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

Additional Key Words and Phrases: Sampled Softmax Loss, Collaborative Filtering, Graph Neural Networks, Long-tail Recommendation

ACM Reference Format:
Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2018. On the Effectiveness of Sampled Softmax Loss for Item Recommendation. *J. ACM* 37, 4, Article 111 (August 2018), 25 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In recent years, studies on recommendation modeling have been extensively conducted. Many model architectures have been proposed to capture user preference from user-item interactions, covering multilayer perceptrons [19], attention mechanisms [18], and graph neural networks [17, 51], etc. However, relatively few studies focus on the learning objective — the way to train and optimize the model parameters. Specifically, most of work casts the item recommendation problem into a supervised learning task, and adopts one of the following learning objectives:

- **Pointwise loss**. From the perspective of binary classification or regression, it treats the observed user-item interactions as positive instances, and other missing data as weak negatives. For a user-item pair, it encourages the model prediction to fit the corresponding label, so as to approach user preference directly. The common choices include binary cross-entropy [19, 40] and mean square error [16, 25].
- **Pairwise loss**. It models the relative ordering of a positive item over an unobserved item for a target user. Numerically, it forces the model to score an observed item higher than its unobserved counterparts. Representative pairwise losses are BPR [37, 44] and WARP [45].
- **Softmax loss**. By applying a softmax function, it normalizes the model predictions over all items as a probability distribution. It then maximizes the probability of the observed items as compared with that of the unobserved items. Although aligned well with the ranking metrics that emphasize the top ranked positions [4, 35], softmax loss is relatively little used in recommender systems. One reason is that softmax loss assumes the probabilities of all items for each user sum up to 1, which may be too computationally expensive to scale to large datasets — in practice, the scale of items easily reaches millions or even larger. Additionally, calculating the exact probability of each possible interaction may be intractable in streaming environments [14, 50] where the complete item space is not always available.

Sampled softmax (SSM) emerges as a substitute for softmax. The basic idea is to use a sampled subset of negatives instead of all items. As such, it not only inherits the desired property of ranking, but also reduces the training cost dramatically [9]. However, directly optimizing classical recommendation models such as MF [26] may not yield satisfactory performance (*cf.* Table 2). Typically, existing recommendation work leverages SSM mainly for two purposes: (1) Approximating the softmax function. Prior work [2] argues that SSM loss is a biased version of full softmax loss. One possible solution is the log$Q$ correction [2], which samples negative instances from the softmax distribution. Some follow-on efforts [1, 3, 28, 34, 43, 48, 50] devise different methods to reduce the sampling bias. (2) Performing contrastive learning. Inspired by its success in CV [8, 12, 42] and NLP [10, 11, 27] domains, researchers are exploring contrastive learning for recommendation [46, 55, 56]. They typically use the InfoNCE loss [42] for the auxiliary task, maximizing the agreement of positive pairs as compared with that of negative pairs. Essentially, InfoNCE is an SSM function, since the observed and unobserved user-item pairs can be viewed as positive and negative instances, respectively.

Nonetheless, only very limited studies [9, 49, 55] utilize SSM as the main learning objective for model training. Even though some recommendation work has used it, they do not explore its

properties, failing to answer the questions "Does SSM suit item recommendation?" and "What are the conceptual advantages of SSM, as compared with pointwise and pairwise losses?".

In this work, we seek to better understand SSM loss for item recommendation. Firstly, we conduct theoretical analyses, identifying three **model-agnostic advantages** of SSM loss:

- **Alleviating popularity bias.** Equipped with in-batch negative sampling, its approximated closed-form solution *w.r.t.* interactions is inversely proportional to item popularity in log scale, naturally suppressing the prediction scores of popular items.
- **Mining hard negative samples.** It can discover hard negative samples, which contribute more informative and larger gradients to the optimization and representation learning.
- **Maximizing the ranking metric.** Optimizing SSM loss is consistent with maximizing the Discounted Cumulative Gain (DCG) metric, which is more suitable for the top-K task.

We further conduct experiments to justify the superiority of SSM for item recommendation. Specifically, among collaborative filtering (CF) family, we select four classical models for studying: matrix factorization (MF) [26] representing ID-based methods, SVD++ [25] being the representation of history-based approaches, while NGCF [44] and LightGCN [17] on behalf of graph-based models, respectively. We have two observations: (1) In both normal and long-tail testing scenarios, history-based and graph-based models achieve leading performance, validating the usefulness of SSM; (2) To our surprise, when applying SSM on MF, we observe large performance degradation in most cases. Scrutinizing these models, we find that MF has no mechanism to adapt the representation magnitude, as compared with SVD++, NGCF, and LightGCN. Hence, we hypothesize that SSM may be good at learning the **representation directions**, but falls short in adjusting the **representation magnitudes**. Moreover, we give theoretical analyses of this hypothesis. In contrast, the history-based and graph-based models can compensate for the weakness of SSM.

In a nutshell, our contributions are summarized as follows:

- We present a deep understanding of SSM and theoretically analyze its model-agnostic advantages: reducing popularity bias, mining hard negatives, and maximizing DCG metric.
- We point out the weakness of SSM loss in learning representation magnitudes, and show that history- and graph- based recommenders compensate for this weakness.
- Extensive experiments on four benchmark datasets justify the rationality of our analyses, demonstrating the superiority of SSM for item recommendation.

2 SSM FOR RECOMMENDATION

Suppose we have a dataset $\mathcal{D}$ that consists of $\mathfrak{M}$ users, $\mathfrak{N}$ items, and $|\mathcal{D}|$ observed interactions. The goal of item recommendation is to recommend a user with a list of items she may be interested in. Noise contrastive estimation (NCE) [15] treats this as a binary classification task, discriminating between the observed data and some artificially generated noises. Specifically, for each observed interaction (u, i), where item i is in user u's set of adopted items $\mathcal{P}_u$, we independently sample N negative items $\mathcal{N} = \{j_1, j_2, \cdots, j_N\}$ from a noise distribution p_n. Let $\mathcal{I} = \{i\} \cup \mathcal{N}$ be the union of the positive item and the sampled negatives. We assign a binary label C_k to each tuple $(u, k) \in u \times \mathcal{I}$, where $C_k = 1$ if $k = i$ and 0 otherwise. The priors are $p(C_k = 1) = 1/(1 + N)$ and $p(C_k = 0) = N/(1 + N)$. Using Bayes' rule, the posterior probabilities are:

$$p(C_k = 1|u, k) = \frac{p(u, k|C_k = 1)}{p(u, k|C_k = 1) + Np(u, k|C_k = 0)}, \tag{1}$$

$$p(C_k = 0|u, k) = \frac{Np(u, k|C_k = 0)}{p(u, k|C_k = 1) + Np(u, k|C_k = 0)}. \tag{2}$$

Without loss of generality, let $p(C_k = 1|u, k) = \sigma(f(u, k))$, where $\sigma(\cdot)$ is the *sigmoid* function, and $f(u, k)$ measures the affinity score between u and k. As such, we can obtain:

$$p(u, k|C_k = 1) = Np(u, k|C_k = 0) \exp\left(f(u, k)\right). \tag{3}$$

Note that among the $(N + 1)$ tuples, only one is labeled 1. Our goal is to maximize the following joint conditional probability:

$$\begin{aligned} P &\overset{(i)}{=} \frac{p(u, i, j_1, \cdots, j_N|C_i = 1, C_{j_1} = 0, \cdots, C_{j_N} = 0)}{\sum_{j\in\mathcal{I}} p(u, i, j_1, \cdots, j_N|C_i = 0, C_{j_1} = 0, \cdots, C_j = 1, \cdots, C_{j_N} = 0)} \\ &\overset{(ii)}{=} \frac{p(u, i|C_i = 1) \prod_{k\in\mathcal{I}\setminus\{i\}} p(u, k|C_k = 0)}{\sum_{j\in\mathcal{I}} p(u, j|C_j = 1) \prod_{l\in\mathcal{I}\setminus\{j\}} p(u, l|C_l = 0)} \\ &\overset{(iii)}{=} \frac{Np(u, i|C_i = 0) \exp\left(f(u, i)\right) \prod_{k\in\mathcal{I}\setminus\{i\}} p(u, k|C_k = 0)}{\sum_{j\in\mathcal{I}} Np(u, j|C_j = 0) \exp\left(f(u, j)\right) \prod_{l\in\mathcal{I}\setminus\{j\}} p(u, l|C_l = 0)} \\ &\overset{(iv)}{=} \frac{\exp\left(f(u, i)\right)}{\sum_{j\in\mathcal{I}} \exp\left(f(u, j)\right)}, \end{aligned} \tag{4}$$

where (i) measures the normalized conditional probability that the observed item i is correctly labeled as 1 while the rest are labeled as 0 (cf. the numerator) against any other situation where some $j \in \mathcal{I}$ is labeled as 1 while the rest are labeled as 0 (cf. the denominator), (ii) follows from the independence assumption of the $(N + 1)$ tuples, (iii) is the direct result of applying Equation (3), and (iv) utilizes the fact that both the numerator and denominator share a common factor of $N \prod_{k\in\mathcal{I}} p(u, k|C_k = 0)$. Usually, the negative-logarithm is applied for numerical stability, resulting in the sampled-softmax loss function:

$$\mathcal{L}_{SSM} = -\frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \log \frac{\exp\left(f(u, i)\right)}{\exp\left(f(u, i)\right) + \sum\limits_{j\in\mathcal{N}} \exp\left(f(u, j)\right)}. \tag{5}$$

Here, we omit the regularization term for simplicity. Typically, we use the mini-batch stochastic gradient descent method, *e.g.,* Adam [24], to optimize model parameters. To make full use of parallel computing of modern hardware, in-batch negative sampling [20] is commonly adopted, that is, treating positive items of other users in the same batch as the negatives. In expectation, this is equivalent to sampling based on the empirical frequency of items in the dataset. Common choices of the affinity function $f(u, i)$ are cosine similarity or inner product of the representations of u and i with a temperature coefficient. Recently, [23, 46] have demonstrated that cosine similarity with temperature coefficient is a better choice, since it endows SSM with the ability to mine hard negatives in a bounded interval. We will go deep into this property later. We should emphasize that throughout this paper, unless otherwise stated, SSM is equipped with temperature-aware cosine similarity function and in-batch negative sampling.

3 THEORETICAL ANALYSES

In this section, we first conduct thorough theoretical analyses to reveal three advantages of SSM for item recommendation. We then identify the potential limitation of SSM in learning representation magnitude, and show that message passing scheme can compensate for this limitation.

3.1 Properties of SSM

3.1.1 ***Alleviating Popularity Bias.*** Intuitively, items with a larger frequency are more likely to be involved in a batch. In other words, this is equivalent to choosing negative items from an empirical frequency sampler, in expectation. As a consequence, popular items are prone to be

penalized as negatives, preventing the model from recommending them to some degree [50]. Here, we prove this statement theoretically.

Let $\mathcal{L}_u$ be the loss on user u and $l(u,i)$ be the individual loss term of positive sample (u,i), then we have,

$$\mathcal{L}_u = \sum_{i \in \mathcal{P}_u} l(u,i) = \sum_{i \in \mathcal{P}_u} \left\{ -\log \frac{\exp\left(f(u,i)\right)}{\exp\left(f(u,i)\right) + \sum_{j \in \mathcal{N}} \exp\left(f(u,j)\right)} \right\}.$$

The gradient of $\mathcal{L}_u$ *w.r.t.* $f(u,i)$ is as follows:

$$\begin{aligned} &\frac{\partial}{\partial f(u,i)} \left\{ \sum_{k \in \mathcal{P}_u} \left[\log \left(1 + \sum_{j \in \mathcal{N}} \exp\left(f(u,j) - f(u,k)\right) \right) \right] \right\} \\ &= \sum_{k \in \mathcal{P}_u} \frac{\partial}{\partial f(u,i)} \log \left[1 + N \mathop{\mathbb{E}}_{j \sim p_n} \exp\left(f(u,j) - f(u,k)\right) \right]. \end{aligned} \tag{6}$$

Here, we use the law of large numbers, changing the sum operation to the expectation. p_n is the distribution of negative samples which is predefined. Note that there are two cases where the term $f(u,i)$ is involved: (1) $k = i$, that is, positive sample is (u,i). Note that with probability $p_n(i)$, item i is sampled as a negative item in this case. As such, the gradient of this part is

$$-1 + \frac{1 + N p_n(i)}{\exp\left(f(u,i)\right) + N \mathbb{E}_{j \sim p_n} \exp\left(f(u,j)\right)} \exp\left(f(u,i)\right). \tag{7}$$

(2) $k \neq i$ but $j = i$, that is, only negative sample is (u,i). In this case, the gradient is

$$\sum_{k \in \mathcal{P}_u \setminus \{i\}} \frac{N p_n(i) \exp\left(f(u,i)\right)}{\exp\left(f(u,k)\right) + N \mathbb{E}_{j \sim p_n} \exp\left(f(u,j)\right)}. \tag{8}$$

By adding (7) and (8) together, we obtain the total gradient of $\mathcal{L}_u$ *w.r.t.* $f(u,i)$. We further enforce the total gradient to be zero and obtain the nearly closed-form solution of $f(u,i)$:

$$f^*(u,i) = \log \frac{N \mathop{\mathbb{E}}_{j \sim p_n} \exp\left(f(u,j)\right)}{1 + N |\mathcal{P}_u| p_n(i)}. \tag{9}$$

Here we use the fact that $\exp\left(f(u,k)\right) \ll N \mathbb{E}_{j \sim p_n} \exp\left(f(u,j)\right), k \in \mathcal{P}_u$ when $N \to \infty$. Notice that the larger $p_n(i)$ is (*i.e.,* the more popular the item is), the smaller $f^*(u,i)$ will be. This is in line with Inverse Propensity Weighting (IPW) methods that adjust the data distribution to be even by reweighting the training instances for bias reduction [6, 39]. Prior work [55] revealed that InfoNCE [42] (which has a similar formula as SSM) and IPW share the same global optima when setting the proposal distribution to be the propensity score. However, to our knowledge, we are the first to derive the closed-form solution of SSM, providing more intuitive explanations for why SSM can alleviate popularity bias and further enabling flexible control over item popularity. We will conduct quantitative experiments to verify the strength of SSM in suppressing popularity bias in Section 4.3.

It is worth mentioning that besides the in-batch negative sampling strategy, one can adopt any other negative sampler to pursue more flexibility. For example, a naive sampler that samples N negatives for each observed interaction independently from any predefined p_n, at the expense of training speed. We will conduct experiments to study the impact of p_n and N in Sections 4.4.2 and 4.4.3, respectively.

3.1.2 ***Hard Negative Mining.*** In [23, 46], the authors analyzed that the InfoNCE loss in self-supervised learning is able to perform hard negative mining. Due to the similar formulae of SSM and InfoNCE, we find that SSM exhibits similar power in mining hard negative items when learning user representation.

Formally, we denote the final representations of user u and item i as $\boldsymbol{z}_u$ and $\boldsymbol{z}_i$ respectively and adopt cosine similarity with temperature coefficient τ to measure the agreement between u and i, that is, $f(u,i) = \frac{\boldsymbol{s}_u^\top \boldsymbol{s}_i}{\tau}$, where $\boldsymbol{s}_u$ and $\boldsymbol{s}_i$ are the normalized representations, *i.e.,* $\boldsymbol{s}_u = \frac{\boldsymbol{z}_u}{\|\boldsymbol{z}_u\|}, \boldsymbol{s}_i = \frac{\boldsymbol{z}_i}{\|\boldsymbol{z}_i\|}$. Then the gradient of $l(u,i)$ *w.r.t.* the user representation $\boldsymbol{z}_u$ is as follows [3]:

$$\frac{\partial l(u,i)}{\partial \boldsymbol{z}_u} = \frac{c(i) + \sum_{j \in \mathcal{N}} c(j)}{\tau \|\boldsymbol{z}_u\|}, \tag{10}$$

where,

$$c(k) = \begin{cases} \left(\boldsymbol{s}_k - (\boldsymbol{s}_u^\top \boldsymbol{s}_k)\boldsymbol{s}_u\right)^\top (P_{uk} - 1), & \text{if } k = i \\ \left(\boldsymbol{s}_k - (\boldsymbol{s}_u^\top \boldsymbol{s}_k)\boldsymbol{s}_u\right)^\top P_{uk}, & \text{if } k = j \end{cases} \tag{11}$$

$$P_{uk} = \frac{\exp(\boldsymbol{s}_u^\top \boldsymbol{s}_k/\tau)}{\sum_{l \in \mathcal{I}} \exp(\boldsymbol{s}_u^\top \boldsymbol{s}_l/\tau)}. \tag{12}$$

Then, we focus on the magnitude contribution of negative item $j \in \mathcal{N}$, *i.e.,* $\|c(j)\|$, which is proportional to the following term:

$$g(x) = \sqrt{1-x^2} \exp\left(\frac{x}{\tau}\right), x \in [-1, 1], \tag{13}$$

where $x = \boldsymbol{s}_u^\top \boldsymbol{s}_j$ denotes the cosine similarity between $\boldsymbol{z}_u$ and $\boldsymbol{z}_j$. By scrutinizing $g(x)$, we have the following insights: (1) when setting a proper τ (empirically in the range of 0.1 to 0.2), hard negative items which have larger x offer larger gradients to guide the optimization, thus making learned representations more discriminative and accelerating the training process. (2) when x becomes large enough (*i.e.,* $x > \left(\sqrt{\tau^2+4} - \tau\right)/2$), $g(x)$ decreases sharply to 0 as $x \to 1$, which indicates that SSM has a nice property that weakens the impact of too-hard negatives since they may be potential positives. It is worth noting that the key difference between our analysis in this work and previous studies [46] is that we specifically focus on the hard negative mining property within user-item interactions. Previous work has mainly centered around self-supervised learning techniques to find hard samples from augmented nodes of the same type (i.e., user or item nodes). In contrast, by exploring the potential of using hard negative mining within the context of user-item interactions, we aim to provide more direct insights into the relationship between users and items, which can improve our understanding of the underlying mechanisms of recommender systems. Compared with other hard negative mining techniques like choosing hard negatives from top recommendation results of the current training state [53] or setting a margin to block the gradient of easy negative items [30], SSM uses a more graceful way through in-batch negative sharing strategy and control factor τ, which scarcely increases training complexity.

3.1.3 ***Maximizing DCG.*** Discounted Cumulative Grain (DCG) is a widely-adopted ranking metric which uses a graded relevance scale to calculate the utility score. The contribution to the utility from a relevant item reduced logarithmically proportional to the position of the ranked list, which mimics the behavior of a user who is less likely to examine items at larger ranking position. Formally, DCG

[3]We adopt the numerator layout notation here.

is defined as follows:

$$DCG(\pi_{f_u}, y) = \sum_{i=1}^{|\mathcal{I}|} \frac{2^{y_i} - 1}{\log_2(1 + \pi_{f_u}(i))}, \tag{14}$$

where π_{f_u} is a ranked list over $\mathcal{I}$ induced by f for user u; y is a binary indicator: $y_i = 1$ if item i has been interacted by u, otherwise $y_i = 0$; $\pi_{f_u}(i)$ is the rank of i. It is worth noting that, in practice, there may be more than one item that has been interacted by u in $\mathcal{I}$. Inspired by [4], we derive the relationship between SSM loss and DCG:

$$\begin{aligned}
\pi_{f_u}(i) &= 1 + \sum_{j \in \mathcal{I} \setminus \{i\}} \mathbb{I}(f(u,j) - f(u,i) > 0) \\
&\leq 1 + \sum_{j \in \mathcal{I} \setminus \{i\}} \exp(f(u,j) - f(u,i)) \\
&= \sum_{j \in \mathcal{I}} \exp(f(u,j) - f(u,i)).
\end{aligned} \tag{15}$$

Based on the fact $\mathbb{I}(x > 0) \leq \exp(x)$, we have:

$$\begin{aligned}
-\log DCG(\pi_{f_u}, y) &\leq -\log \frac{1}{\log\left(1 + \pi_{f_u}(i)\right)} \\
&\leq -\log \frac{1}{\pi_{f_u}(i)} \\
&\leq -\log \frac{\exp(f(u,i))}{\sum_{j \in \mathcal{I}} \exp(f(u,j))}.
\end{aligned} \tag{16}$$

As the fact $\log(1+x) \leq x$, we can safely get that: $-\log DCG$ is a lower bound of SSM, and minimizing SSM is equivalent to maximizing DCG of $\mathcal{I}$. Such finding provides insight that SSM matches well with item recommendation in terms of optimization goal, thus being suitable for optimizing the recommendation models.

3.2 Potential Limitations of SSM

3.2.1 ***SSM Fails in Learning Representation Magnitude.*** In the above section, we have proven that SSM has three desirable properties for item recommendation. Consequently, a spontaneous question is: "Can SSM lead to empirical performance gain across different recommender models?" Towards answering this question, we select three categories of CF methods — ID-, history-, and graph- based methods — as backbone models to justify its superiority. The detailed results are presented in Section 4.2.2. To our surprise, in spite of significant performance gains on both history-based (*i.e.,* SVD++ [25]) and graph-based models (*i.e.,* NGCF [44] and LightGCN [17]), we observe large performance degradation on ID-based model (*i.e.,* MF [37]). Such inconsistency drives us to probe the failure of SSM. Scrutinizing these recommender models, we find that the key lies in adapting representation magnitudes during representation learning. For SSM, due to the adoption of cosine similarity, the representation magnitudes will not affect the optimization process (we ignore the regularization term here), thus SSM will not guide the representation magnitude learning. What's worse, ID-based model (*i.e.,* MF) also has no explicit mechanism to adapt the representation magnitude, resulting in poor quality of representations. On the contrary, as we will prove later, message-passing-based models such as LightGCN can intrinsically adjust the representation magnitude according to the node degree, thus compensating for the limitation of SSM. Hence, we argue that SSM may be good at learning the **representation directions**, but falls short in adjusting the **representation magnitudes**.

3.2.2 ***Message Passing can Adjust the Representation Magnitude.*** Without loss of generality, we define the message passage rule in user-item interaction graph as follows:

$$\boldsymbol{q}_i^{'} = \sum_{u \in \mathcal{P}_i} \frac{1}{|\mathcal{P}_i|^{\alpha_0} |\mathcal{P}_u|^{\alpha_1}} \boldsymbol{p}_u, \tag{17}$$

where α_0 and α_1 are the normalization factors. In this work, we only analyze from the item side, since the user side can be derived in the same way.

Suppose the degree of user node $|\mathcal{P}_u|$ and item node $|\mathcal{P}_i|$ follow the Pareto distribution [32] with parameter $(\alpha, x_m = 1)$, where α is the shape parameter, x_m is the minimum possible value of node degree (we set it to 1). $\boldsymbol{p}_u^{(0)}$ is initialized with some distribution $\mathcal{D}(\mu_0, \sigma_0^2)$, that is

$$\begin{aligned} \mathbb{E}(|\mathcal{P}_u|) = \mathbb{E}(|\mathcal{P}_i|) &= \frac{\alpha}{\alpha - 1}, \\ \mathbb{V}(|\mathcal{P}_u|) = \mathbb{V}(|\mathcal{P}_i|) &= \frac{\alpha}{(\alpha-1)^2(\alpha-2)} \\ \mathbb{E}(\boldsymbol{p}_u) = \mathbb{E}(\boldsymbol{q}_i) &= \mu_0, \\ \mathbb{V}(\boldsymbol{p}_u) = \mathbb{V}(\boldsymbol{q}_i) &= \sigma_0^2 \end{aligned} \tag{18}$$

where $\mathbb{E}(\cdot)$ and $\mathbb{V}(\cdot)$ are the expectation and variance, respectively. Therefore, for a given item i, the expectation of the square of its magnitude $\boldsymbol{q}_i^{'}$ is

$$\mathbb{E}((\boldsymbol{q}_i^{'})^2) = \frac{1}{|\mathcal{P}_i|^{2\alpha_0}} \left\{ \sum_{u \in \mathcal{P}_i} V_u + \left[\sum_{u \in \mathcal{P}_i} E_u \right]^2 \right\}, \tag{19}$$

where,

$$V_u = \mathbb{V}\left(\frac{1}{|\mathcal{P}_u|^{\alpha_1}} \boldsymbol{p}_u \right), \quad E_u = \mathbb{E}\left(\frac{1}{|\mathcal{P}_u|^{\alpha_1}} \boldsymbol{p}_u \right) \tag{20}$$

Since $|\mathcal{P}_u|$ and $\boldsymbol{p}_u$ are independent variables, we have

$$\begin{aligned} V_u &= \mathbb{E}\left(\frac{1}{|\mathcal{P}_u|^{2\alpha_1}} \right) \mathbb{E}\left(\boldsymbol{p}_u^2 \right) - \left[\mathbb{E}\left(\frac{1}{|\mathcal{P}_u|^{\alpha_1}} \right) \right]^2 \left[\mathbb{E}(\boldsymbol{p}_u) \right]^2 \\ &= \frac{\alpha}{\alpha + 2\alpha_1} \left(\sigma_0^2 + \mu_0^2 \right) - \left(\frac{\alpha}{\alpha + \alpha_1} \right)^2 \mu_0^2, \end{aligned} \tag{21}$$

$$\begin{aligned} \sum_{u \in \mathcal{P}_i} E_u &= \sum_{u \in \mathcal{P}_i} \mathbb{E}\left(\frac{1}{|\mathcal{P}_u|^{\alpha_1}} \right) \mathbb{E}\left(\boldsymbol{p}_u \right) \\ &= |\mathcal{P}_i| \frac{\alpha}{\alpha + \alpha_1} \mu_0. \end{aligned} \tag{22}$$

Finally, we can obtain

$$\mathbb{E}\left((\boldsymbol{q}_i^{'})^2 \right) = a |\mathcal{P}_i|^{1-2\alpha_0} + b |\mathcal{P}_i|^{2-2\alpha_0} \tag{23}$$

where,

$$\begin{aligned} a &= \frac{\alpha}{\alpha + 2\alpha_1} \left(\sigma_0^2 + \mu_0^2 \right) - \left(\frac{\alpha}{\alpha + \alpha_1} \right)^2 \mu_0^2 > 0 \\ b &= \frac{\alpha^2 \mu_0^2}{(\alpha + \alpha_1)^2} > 0. \end{aligned} \tag{24}$$

Specially, when $\alpha_0 = \alpha_1 = 0.5$, *i.e.,* the case of LightGCN, $\mathbb{E}\left((\boldsymbol{q}_i')^2\right)$ monotonically linearly increases as $|\mathcal{P}_i|$ increases. We conduct empirical experiments to study the impact of different message-passing strategies (*i.e.,* different α_0 and α_1) in section 4.4.6. While we find an effective way to compensate for the limitation of SSM loss in learning magnitude, the optimal solution is still an open question.

4 EXPERIMENTS

We conduct extensive experiments and answer the following research questions:

- **RQ 1:** Does SSM suit well for item recommendation?
- **RQ 2:** How does SSM perform *w.r.t.* long-tail recommendation, as compared with the existing losses?
- **RQ 3:** How do different components affect SSM?

4.1 Experimental Setup

4.1.1 ***Compared Losses.*** To justify the superiority of SSM on item recommendation, we compare it with diverse losses:

- BCE Loss: A widely used pointwise loss that is formulated as:

$$\mathcal{L}_{BCE} = -\sum_{(u,i)\in\mathcal{D}} \log \sigma(f(u,i)) - \sum_{(u,j)\in\mathcal{D}^-} \log \hat{\sigma}(f(u,j)), \tag{25}$$

  where $\hat{\sigma}(x) = 1 - \sigma(x)$, $\mathcal{D}^-$ is the sampled subset of unobserved interactions.
- BPR Loss: The standard objective function in vanilla NGCF and LightGCN, which encourages the prediction of positive item to be larger than its negative counterpart. The formal formulation is defined as follows:

$$\mathcal{L}_{BPR} = -\frac{1}{|\mathcal{D}|}\sum_{u=1}^{\mathfrak{M}} \sum_{i\in\mathcal{P}_u,\ j\notin\mathcal{P}_u} \log \sigma(f(u,i) - f(u,j)). \tag{26}$$

  where $\mathfrak{M}$ is the total number of users in the dataset.
- SM Loss: It is short for softmax loss which maximizes the probability of the observed items normalized over all items by softmax function, that is:

$$\mathcal{L}_{SM} = -\frac{1}{|\mathcal{D}|}\sum_{(u,i)\in\mathcal{D}} \log\left\{\frac{\exp\left(f(u,i)\right)}{\sum_{j=1}^{\mathfrak{N}}\exp\left(f(u,j)\right)}\right\}, \tag{27}$$

  where $\mathfrak{N}$ is the total number of items in the dataset.
- CCL Loss [30]: It's a contrastive loss proposed recently by maximizing the cosine similarity of positive pairs while minimizing the similarity of negative pairs below a certain margin:

$$\mathcal{L}_{CCL} = -\frac{1}{|\mathcal{D}|}\sum_{(u,i)\in\mathcal{D}}\left[1 - f(u,i) + \frac{w}{|\mathcal{N}|}\sum_{j\in\mathcal{N}}(f(u,j) - m)_+\right] \tag{28}$$

  where $(\cdot)_+$ indicates $\max(0,\cdot)$, $\mathcal{N}$ is a set of randomly sampled negative samples, m is the margin for similarity score of negative samples, w is a hyper-parameter to balance the loss terms of positive samples and negative samples.

We implement SSM and all compared losses on different categories of CF models, ranging from ID-based (*i.e.,* MF [37]) that directly projects the single ID of a user/item into a latent embedding, to history-based (*i.e.,* SVD++ [25]) that takes into consideration the user's historical interactions

Table 1. Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,831	40,841	1,666,869	0.00128
Amazon-Book	52,643	91,599	2,984,108	0.00062
Alibaba-iFashion	300,000	81,614	1,607,813	0.00007

for user representation learning, to graph-based methods (*i.e.,* NGCF [44] and LightGCN [17]) that achieve state-of-the-art performance by performing graph convolutions on the user-item graph.

4.1.2 ***Datasets and Evaluation Metrics.*** We conduct experiments on four benchmark datasets: Gowalla [17, 44], Yelp2018 [17, 44], Amazon-Book [17, 44] and Alibaba-iFashion [46]. Following [17, 44], we use the same 10-core setting for the first three datasets. For Alibaba-iFashion, as processed in [46], we randomly sample 300k users and collect all their interactions over the fashion outfits. The statistics of all four datasets are summarized in Table 1. We follow the same strategy described in [44] to split the interactions into training, validation and test set with a ratio of $7:1:2$. For users in the test set, we follow the all-ranking protocol [17, 44] to evaluate the top-K recommendation performance and report the average Recall@20 and NDCG@20.

4.1.3 ***Hyper-parameter Settings.*** For fair comparison, all methods are trained from scratch which are initialized with Xavier [13]. In line with NGCF and LightGCN, we fix the embedding size to 64 and optimize all models via Adam [24] with the default learning rate of 0.001 and default mini-batch size of 2048. The L_2 regularization term is added, with the coefficient in the range of $\{1e^{-6}, 1e^{-5}, \cdots, 1e^{-1}\}$. The normalization factor of SVD++ is searched in $\{0, 0.5, 1.0\}$. For graph-based method, *i.e.,* NGCG [44] and LightGCN [17], the number of GCN layers K is searched in the range of $\{1, 2, 3, 4\}$. The dropout ratio of NGCF is in $\{0.0, 0.1, , 0.8\}$. The layer combination coefficient of LightGCN is set to $\frac{1}{K+1}$. For BCE, we randomly sample nonobserved interactions to form the negative set $\mathcal{D}^-$ in each training epoch, the ratio of positive to negative is set to $1:4$. For BPR, we randomly sample a noninteracted item of the user as negative for each observed interaction. For SM, we examine both cosine similarity and inner product similarity between user and item representations, and report the best performance. For CCL, as suggested in the paper [30], we tune the margin m among $0.1 \sim 1.0$ at an interval of 0.1 and weight w in the range of $\{1, 150, 300, 1000\}$. The number of negative samples is searched from 1 to 2048. While for SSM, we find cosine similarity always leads to better performance. Since the temperature coefficient τ in both SM loss and SSM is of great importance [46], we use the following strategy to tune it: we first perform grid search in a coarse grain range of $\{0.1, 0.2, 0.5, 1.0\}$, then in a finer grain range, which is based on the result of coarse tuning. For example, if the model achieves the best performance when $\tau = 0.2$ in the first stage, then we tune it in the range of $\{\cdots, 0.16, 0.18, 0.22, 0.24 \cdots\}$ at finer granularity. The overall results reported in Table 2 are the average value of 5 runs.

4.2 Overall Performance Comparison

4.2.1 ***Comparison Among Different Losses.*** Comparing the best performance each loss can achieve in Table 2, we have the following observations:

- Among the compared losses, BCE performs worst on all four datasets, which suggests the limitations of pointwise objective. Specifically, as it approaches the task as a binary classification machine learning task, fitting the exact value of labels may introduce noise in item recommendation. BPR performs better than BCE in most cases, verifying the superiority of pairwise

Table 2. Performance of different recommenders under different loss functions. The bold indicates the best result for each recommender on each dataset. The superscript $*$ indicates the best result on each dataset.

Dataset		Gowalla		Yelp2018		Amazon-Book		Alibaba-iFashion	
Recommender	**Loss**	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF	**BCE**	0.1555	0.1294	0.0571	0.0464	0.0303	0.0236	0.0950	0.0439
	BPR	0.1558	0.1254	0.0562	0.0454	0.0352	0.0266	0.1031	0.0483
	SM	0.1777	0.1434	**0.0709**	**0.0581**	0.0515	0.0399	0.1225	0.0593
	CCL	**0.1837**	**0.1493**	0.0698	0.0572	**0.0559**	**0.0447**	**0.1229**	**0.0585**
	SSM	0.1231	0.0878	0.0509	0.0404	0.0473	0.0367	0.0841	0.0400
SVD++	**BCE**	0.1549	0.1284	0.0564	0.0459	0.0306	0.0235	0.0969	0.0458
	BPR	0.1589	0.1302	0.0569	0.0458	0.0373	0.0286	0.1094	0.0509
	SM	**0.1852**	**0.1550**	0.0679	0.0555	0.0478	0.0371	0.1157	0.0559
	CCL	0.1819	0.1453	0.0693	0.0567	**0.0557**	**0.0440**	0.1233	0.0591
	SSM	0.1669	0.1376	**0.0693**	**0.0569**	0.0547	0.0429	**0.1276**	**0.0622**
NGCF	**BCE**	0.1541	0.1303	0.0563	0.0463	0.0330	0.0264	0.0547	0.0238
	BPR	0.1548	0.1248	0.0579	0.0477	0.0357	0.0273	0.0923	0.0416
	SM	0.1766	0.1471	0.0699	0.0573	0.0486	0.0377	0.1169	0.0552
	CCL	0.1778	0.1411	0.0651	0.0535	0.0516	0.0406	0.1068	0.0507
	SSM	**0.1854**	**0.1548**	**0.0736**	**0.0608**	**0.0552**	**0.0431**	**0.1296***	**0.0629***
LightGCN	**BCE**	0.1743	0.1491	0.0628	0.0515	0.0373	0.0292	0.0983	0.0458
	BPR	0.1824	0.1554	0.0640	0.0524	0.0417	0.0322	0.1086	0.0511
	SM	0.1756	0.1429	0.0708	0.0580	0.0489	0.0377	0.1155	0.0558
	CCL	0.1790	0.1407	0.0669	0.0554	0.0528	0.0416	0.1203	0.0570
	SSM	**0.1869***	**0.1571***	**0.0737***	**0.0609***	**0.0590***	**0.0459***	**0.1253**	**0.0599**

objectives to capture relative relations among items. Both SM and CCL have significant gains over BCE and BPR on Yelp2018, Amazon-Book, and Alibaba-iFashion datasets. A common point of SM and CCL is that they both compare one positive sample with multiple negative samples. This suggests that enlarging the number of negative samples during training is beneficial to representation learning. On Gowalla, SM and CCL have on-par performance compared to BPR.

- The best performance on each dataset is always achieved by SSM, empirically verifying the advantages of SSM for item recommendation. We attribute the gains to: (1) the use of multiple negatives for each observed interaction at every iteration, as compared with pointwise and pairwise loss. In addition, with a proper temperature coefficient, SSM benefits from dynamic hard negative mining (see Section 3.1.2), in which the hard negative items offer larger gradients to guide the optimization. Moreover, SSM has strong connections with DCG ranking metric as we have analyzed in Section 3.1.3 — meaning that SSM directly optimizes the goal of model evaluation. (2) using only a fraction of items from the whole item set for optimization, as compared with SM, which greatly alleviates the training cost and difficulty, especially when the set of whole items becomes extremely large. Evidence supporting this assertion is that the improvements of SSM over SM become more significant on larger datasets (*i.e.,* Yelp2018, Amazon-Book, and Alibaba-iFahsion) than a smaller one (*i.e.,* Gowalla). Another possible reason lies in that by introducing randomness, SSM can avoid the phenomenon that a few extremely hard negative items dominate the optimization in SM.

4.2.2 ***Impact of Different Recommenders.*** In the previous section, we have verified the superiority of SSM loss compared with other losses. Here, we study if SSM loss is consistently a good choice for optimizing different CF models. From Table 2, we can find that:

- Paired with SSM, both history-based and graph-based methods achieve leading performance. This again verifies the superiority of SSM for item recommendation. However, when implemented on MF, SSM performs poorly. We attribute this phenomenon to the following two causes: (1) as analyzed in Section 3.2, SSM neglects the impact of representation magnitudes when calculating matching score since it adopts cosine similarity function. Worse still, MF merely maps an ID into an embedding vector, without explicit design to adjust the magnitude of representations. As such, the combination of MF and SSM cannot compensate for the flaws in adjusting the magnitude of representations and hence leads to low-quality representation. (2) as analyzed in Section 3.1.1, SSM will penalize the predicted score for popular items. However, prior work [54] has demonstrated that popularity bias can be somehow good if properly leveraged. Given that MF has no mechanism to leverage popularity information, the combination of MF and SSM will under-estimate the popular items, leading to undesirable performance. In contrast, we have proven that message-passing methods inherently are capable of adjusting the representation magnitude based on node degree. Therefore, paired with a message-passing method, SSM will lead to excellent performance.
- Focusing on BPR or SSM only, we find that as the recommender becomes more complicated, the performance improves gradually. This is in line with the intuition that adding GCN layers will capture higher-order collaborative signal which is of benefit to recommendation. However, as we can see, the combination of SVD++ and SSM outperforms the combination of LightGCN and BPR. This suggests that BPR is less effective to mine user preference underlying observed interactions. In contrast, the performance gain provided by SSM could even be larger than that from adding GCN layers, which further justifies its effectiveness.
- Between the two graph-based recommenders, NGCF is less competitive than LightGCN when paired with traditional losses, *e.g.,* BCE and BPR. Prior studies typically attribute this to the adoption of multiple nonlinear transformations, which increases the difficulties to train the recommender well [7, 17]. However, armed with SSM loss, NGCF achieves an equivalent level (*e.g.,* on Gowalla and Yelp2018 datasets) or even better (*e.g.,* on Alibaba-iFashion dataset) performance compared with LightGCN. This indicates that SSM promotes the training process of more complicated models, showing great potential for solving complex optimization problems.

4.3 Long-tail Recommendation

As analyzed in section 3.1.1, SSM is promising for alleviating the popularity bias. To verify this property, we follow [46], splitting items into ten groups based on the frequency while keeping the total number of interactions of each group the same. The items in groups with larger GroupIDs have larger degrees. As such, we decompose the Recall@20 metric of the whole dataset into ten parts, each of which represents the contribution of a single group as follows:

$$\text{Recall} = \frac{1}{\mathfrak{M}} \sum_{u=1}^{\mathfrak{M}} \frac{\sum_{g=1}^{10} \left| (l_{rec}^u)^{(g)} \cap l_{test}^u \right|}{\left| l_{test}^u \right|} = \sum_{g=1}^{10} \text{Recall}^{(g)} \tag{29}$$

where $\text{Recall}^{(g)}$ measures the recommendation performance over the g-th group, l_{rec}^u and l_{test}^u are the items in the top-K recommendation list and relevant items for user u, respectively. We report the results in Fig. 1 and have the following observations:

- Implemented on LightGCN, BCE, BPR, and SM all show a strong inclination to recommend high-degree items, while leaving less-popular relevant items seldom exposed. We should

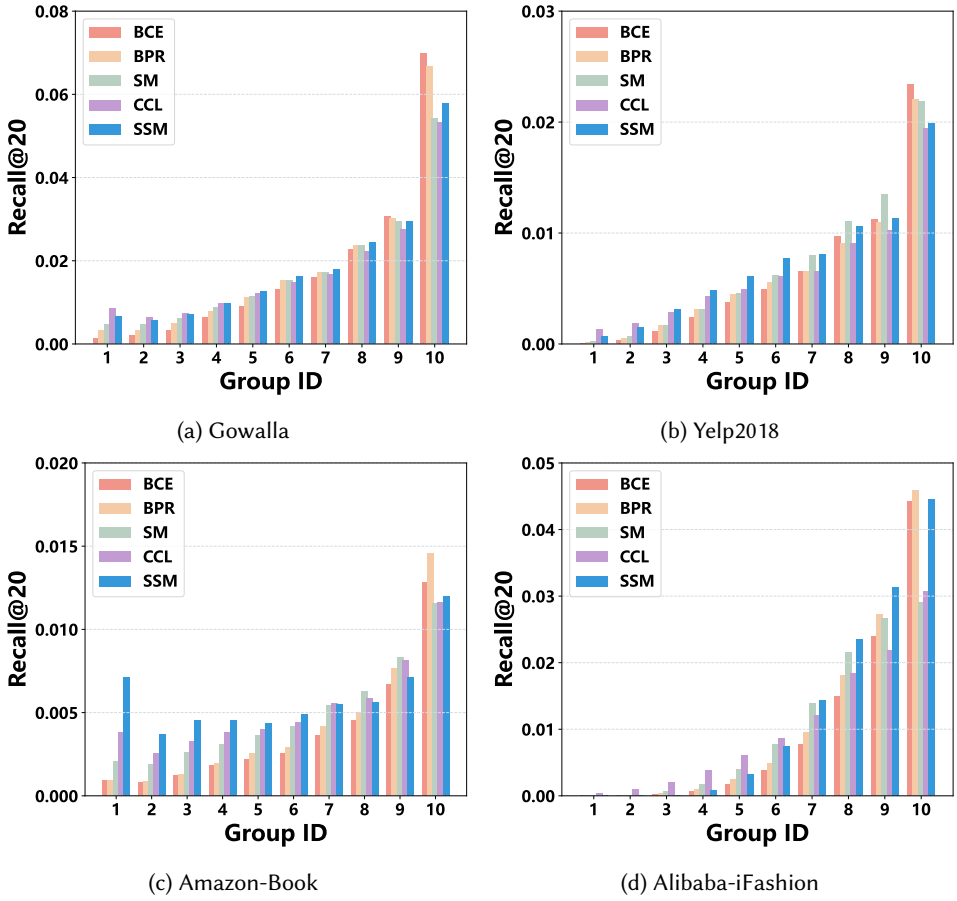

Fig. 1. Performance comparison over different item groups among different objectives.

emphasize that the most popular group, *i.e.,* the 10-th group, only contains less than 1% of item spaces (0.65%, 0.83%, 0.83%, 0.22%, respectively) but contributes more than 35% of the total Recall scores on four datasets. On the contrary, the least popular group, *i.e.,* the 1-st group, contains most of the item spaces (25.51%, 35.70%, 31.82%, 64.43%, respectively) but contributes less than 3% of the total Recall scores. This indicates that paired with LightGCN, BCE, BPR, and SM hardly learn high-quality representations for long-tail items. As such, the recommendation models suffer from popularity bias. What's worse, the feedback loop of recommender system will further intensify popularity bias over time, resulting in the Matthew effect [6].

- CCL and SSM perform well on long-tail groups (those with smaller group ID), showing the potential in alleviating popularity bias. This is consistent with our analysis in Section 3.1.1. Making comparisons between CCL and SSM, we find CCL performs better in the groups with the smallest group ID on three out of four datasets. In contrast, SSM exhibits stronger competitiveness in the waist and head groups. These admit that SSM balances well on normal and long-tail recommendation tasks, that is, SSM can not only promote the exposure of less popular items but also guarantee overall performance. Surprisingly, for SSM, the contribution

Table 3. Impact of the in-batch negative sampling strategy. The backbone model is LightGCN.

Dataset	Gowalla		Yelp2018		Amazon-Book		Alibaba-iFashion	
Loss	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
BCE-IB	0.1429	0.1196	0.0524	0.0427	0.0361	0.0269	0.1107	0.0526
BPR-IB	0.1225	0.0877	0.0492	0.0381	0.0470	0.0364	0.0937	0.0449
CCL-IB	0.1093	0.0771	0.0484	0.0377	0.0495	0.0404	0.0644	0.0295
SSM	**0.1869**	**0.1571**	**0.0737**	**0.0609**	**0.0590**	**0.0459**	**0.1253**	**0.0599**

of each group on Amazon-Book is nearly uniformly-distributed. This again justifies that, by suppressing the predicted scores of popular items, the representation learning sheds more light on long-tail items, so as to establish better representations of these items.

- Another interesting observation is that: for long-tail groups, the performance rank of losses typically exhibits the following order: pointwise loss < pairwise loss < setwise loss [4]; however, for the most popular group, the loss order reverses: setwise loss < pairwise loss < pointwise loss. This suggests that the limited expressiveness of binary pointwise loss and pairwise loss cannot fully capture user preference toward items. Instead, they adopt a conservative policy by recommending popular items.

4.4 Component Analysis on SSM

In this section, we move on to studying different designs in SSM. We first investigate the rationality of in-batch negative sampling strategy in SSM by making comparisons among different losses that are also equipped with in-batch negative sampling strategy. Then the influences of the negative sampling distribution and the number of negative samples are studied. After that, we present the impact of different similarity measurements during model training and testing. Furthermore, we conduct empirical studies on the normalization factors in message passing rules. Finally, we numerically compare the training time of different objectives.

4.4.1 ***Impact of In-batch Negative Sampling Strategy.*** By default, SSM utilizes the in-batch negative sampling strategy to make full use of parallel computing of modern hardware. Indeed, most sampling-based approaches (*e.g.,* BCE, BPR, CCL) can also be equipped with this acceleration technique. For example, we can extend BCE by assigning label '0' to other items within the same mini-batch, compared with label '1' for the corresponding positive item, termed 'BCE-IB'. Similarly, we name the multiple-negative version of BPR with in-batch sharing strategy as 'BPR-IB'. For CCL, instead of sampling a set of individual negative items for each observed interaction, we simply regard other items within the same mini-batch as the set of negative items, termed 'CCL-IB'. To determine the influences and the underlying mechanism of the in-batch negative sampling strategy, we compare SSM with the foregoing variants of different losses on all four datasets. Table 3 records the overall performance of each loss on each dataset. Fig. 2 shows the performance discrepancy of group-wise Recall@20 metric (*cf.* Equation (29)) between the vanilla loss and its in-batch version. We have the following observations:

- Jointly analyzing Table 2 and Table 3, we find significant performance drop for BCE-IB, BPR-IB, and CCL-IB on all datasets, except BPR-IB on Amazon-Book. This admits that not every objective function can benefit from in-batch negative sampling strategy. SSM outperforms all compared approaches across the board, verifying the rationality and effectiveness of utilizing in-batch negative sampling strategy for SSM.

[4]We name the groups of SSM, SM, and CCL as the setwise loss, as they all account for a set of user-item interactions

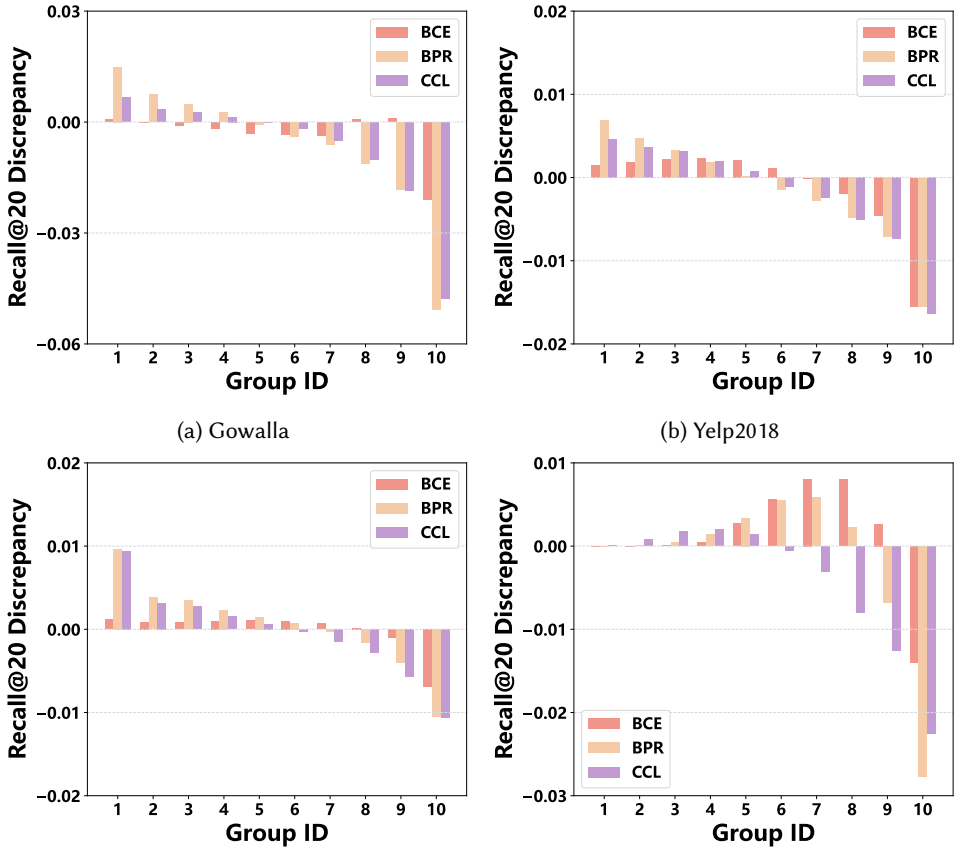

Fig. 2. Performance discrepancy of different item groups among different objectives.

- Fig. 2 provides clues to explain the performance fluctuations of these compared approaches. Specifically, when equipped with in-batch negative sampling strategy, all approaches exhibit better long tail recommendation accuracy, that is, for long-tail groups (groups with smaller ID) in Fig. 2, a consistent improvement *w.r.t.* recall@20 is achieved over the vanilla sampling strategy. On the contrary, the head groups suffer from serious performance deterioration. This verifies that in-batch sampling strategy has the potential to alleviate the popularity bias by over penalizing the prediction score of popular items, which is consistent with our analysis in section 3.1.1. However, this sampling strategy cannot guarantee the overall recommendation accuracy, which will undoubtedly limit its practical value. In contrast, as we have analyzed in section 3.1.3, SSM is highly consistent with the ranking metrics, therefore can well balance the performance of normal recommendation and long-tail recommendation.

4.4.2 ***Impact of Negative Sampling Distribution.*** As SSM is a sampling-based loss, the distribution of negative samples matters for ranking performance. Since the default in-batch negative sampling of SSM is equivalent to sampling negative items from an empirical frequency sampler in expectation, we here introduce a variant of in-batch negative sampling to study the impact of

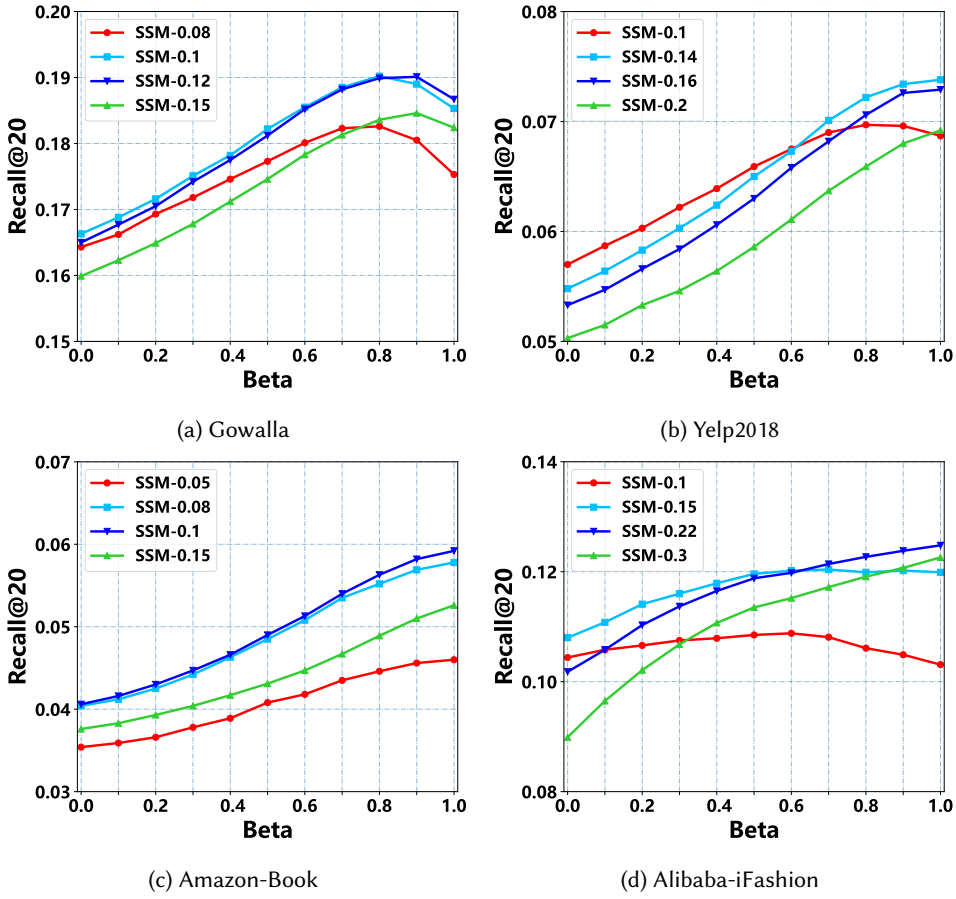

Fig. 3. Impact of negative sampling distribution β and temperature coefficient τ.

negative sampling distribution. Specifically, we use a customized negative sampler as the naive sampler does, but share the sampled N negative samples across the positive pairs in the current batch as the in-batch negative sampling does. For ease of implementation, we let $p_n(i) = f_i^\beta$, which balances between a uniform sampler and a frequency sampler, controlled by coefficient $\beta \in [0, 1]$. Other choices of p_n are left as future work.

We record the fluctuation in Recall@20 as we change β from 0 to 1 at an interval of 0.1, as shown in Fig. 3. We use "SSM-0.1" to represent the temperature coefficient τ of SSM is set to 0.1, and similar notations for others. We find that:

- With the increase in τ, the ranking metric shows a rising trend in general. This admits that the negative sampling distribution indeed influences representation learning. Moreover, a larger β usually implies harder negatives since popular items are sampled with a higher probability. This also justifies the necessity of hard negative mining.
- In most cases, the best performance is achieved at $\beta = 1.0$, which supports the adoption of in-batch negative sampling strategy due to their equivalence in expectation. However, it is still possible to further improve the performance by fine-grained tuning the value of β. For

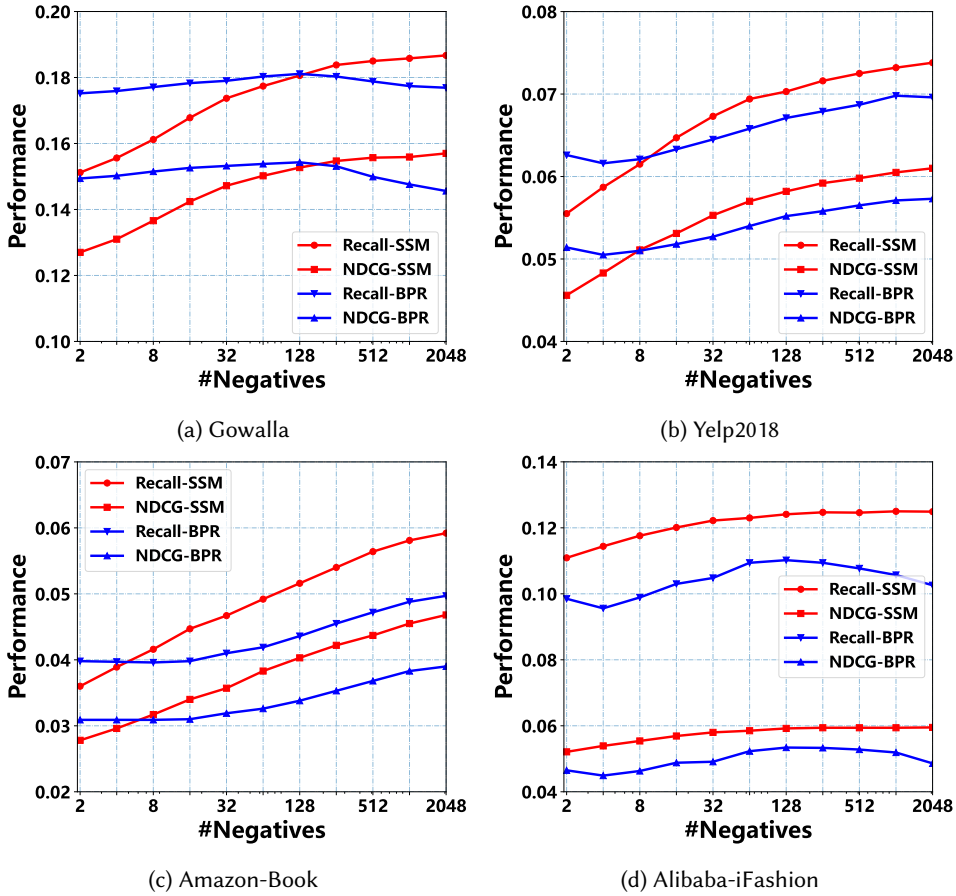

Fig. 4. Impact of negative sampling number N

example, we can obtain a better performance when setting β to 0.9 on Gowalla, compared to the results shown in Table 2.

- Comparing different curves, we can see that when fixing τ to a proper value (*e.g.,* 0.1 on Amazon-Book), the representation learning benefits from hard negative mining as we analyzed in Section 3.1.2. In contrast, a too large (or too small) value of τ, for example, 0.15 (or 0.05) on Amazon-Book, will lead to performance degradation as it's difficult to distinguish hard negative samples from easy ones (or as few too hard negatives dominate the gradient), making the learned representation less quality.

4.4.3 ***Impact of Negative Sampling Number.*** Fig. 4 shows the impact of N, the number of negative samples, using the variant of SSM introduced in the previous section. Here, we extend BPR by using multiple non-interacted items as negative samples. As such, the main difference between BPR and SSM is: BPR uses inner product to measure similarity, while SSM uses temperature-aware cosine similarity. We can see that:

- On Yelp2018 and Amazon-Book datasets, BPR enjoys the benefits of increasing the number of negative samples. This is in line with our intuition that seeing more negative samples

Table 4. Performance comparison among different similarity function combinations during training and testing phases. 'IP' indicates 'Inner Product' similarity while 'Cos' indicates 'Cosine' similarity. We report Recall@20 metric on four datasets.

Similarity Combination	Combination		IP-IP	IP-Cos	Cos-IP	Cos-Cos
	training	IP	✓	✓		
		Cos			✓	✓
	testing	IP	✓		✓	
		Cos		✓		✓
Performance	Gowalla	Recall	0.1085	0.0825	**0.1867**	0.1140
		NDCG	0.0768	0.0551	**0.1567**	0.0794
	Yelp2018	Recall	0.0468	0.0317	**0.0737**	0.0477
		NDCG	0.0369	0.0241	**0.0609**	0.0371
	Amazon-Book	Recall	0.0435	0.0358	**0.0590**	0.0491
		NDCG	0.0336	0.0272	**0.0459**	0.0381
	Alibaba-iFashion	Recall	0.0749	0.0236	**0.1253**	0.0629
		NDCG	0.0357	0.0095	**0.0599**	0.0269

during model training makes a more sophisticated recommender. However, we also observe performance degeneration when N exceeds some threshold, say 128 on Gowalla and Alibaba-iFashion. The possible reasons are two-fold: (1) increasing the size of negative samples will inevitably add difficulties to model optimization, especially when the supervision signal is highly sparse, *e.g.,* on Alibaba-iFahsion. (2) BPR regards all negative samples as equally important, regardless of their hardness, which leads to less-information gradients at training step [36]. Another drawback of the generalized BPR we should emphasize is its training efficiency. We can see later from Table 6 that the generalized BPR is more than 50x slower than SSM for each training epoch when $N = 2048$ on Amazon-Book.

- The performance of SSM keeps getting better as N increases on all four datasets. Moreover, when N becomes sufficient (*e.g.,* $N > 8$ on Amazon-Book), SSM surpasses BPR, even though the total number of negative samples is N in a mini-batch, in contrast to $|\mathcal{B}| \times N$ of BPR, where $|\mathcal{B}|$ is the size of mini-batch. Moreover, as N increases, the performance gap becomes larger. We attribute this to the advantage of SSM in mining hard negative samples as analyzed in Section 3.1.2.

4.4.4 ***Impact of Different Similarity Measurements.*** We find an interesting phenomenon of SSM, which supports our claims about the necessity of adapting representation magnitudes during representation learning. Specifically, we use LightGCN as backbone model for SSM, and adopt different similarity measurement combinations for model training and testing — we use either 'Inner Product' similarity or 'Cosine' similarity in SSM during model training, and use one of them to generate predictions during testing, resulting in four different combinations. The empirical results are shown in Table 4. We find that:

- Cosine similarity is a better choice than Inner Product similarity for SSM during training. This is consistent with our analysis in Section 3.1.2: equipped with temperature-aware cosine similarity, SSM is capable of performing hard negative mining, so as to enhance the quality of the learned representations. A nice property of Cosine similarity is that the similarity value is bounded in the interval $[-1, 1]$, making it easier to train.

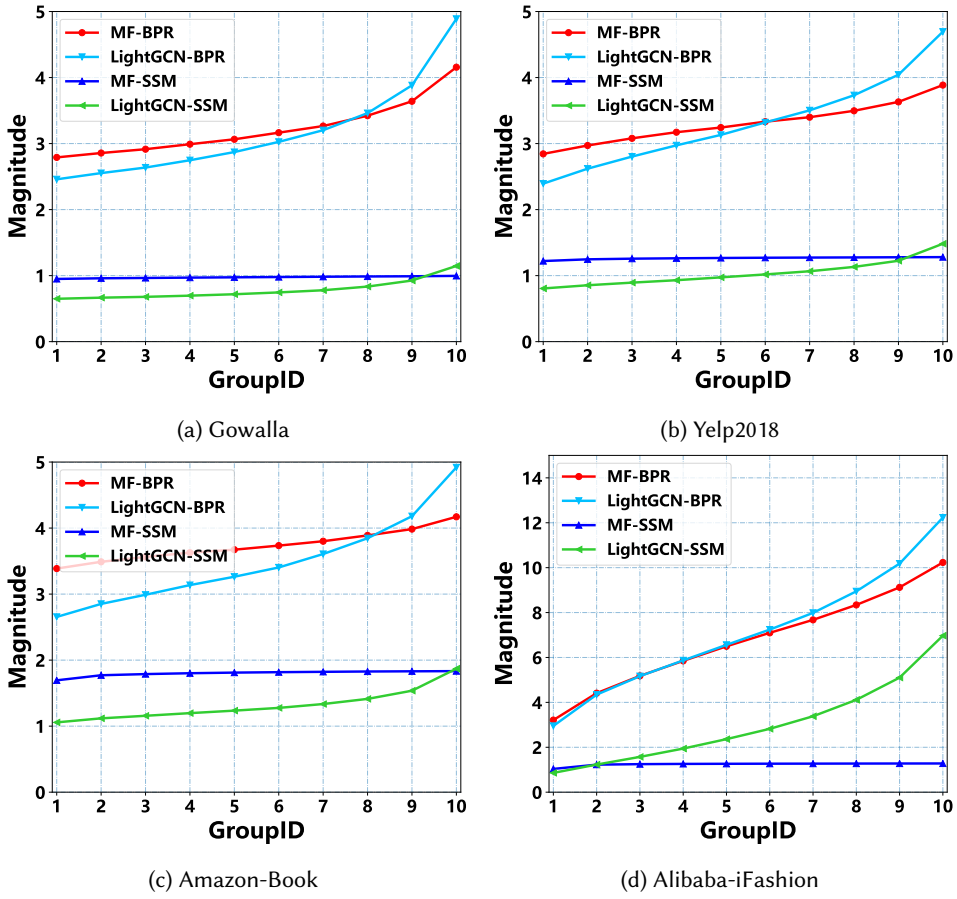

Fig. 5. Empirical study on the magnitude of item representations learned by BPR and SSM.

- Compared with Cosine similarity, Inner Product similarity performs better for SSM during testing. The underlying reason is that in addition to the angle between user and item representations, Inner Product similarity additionally considers the magnitude of two representations, thus making full use of the learned representations for prediction.

4.4.5 ***Empirical study on magnitude of item representations.*** By default, SSM uses the cosine function to measure user-item similarity, which implies that item magnitudes do not contribute to the loss and cannot guide magnitude learning. To provide empirical evidence for this claim, we conducted a study comparing the item representation magnitudes learned by SSM and BPR, using MF and LightGCN as backbones, creating four models: 'MF-SSM', 'LightGCN-SSM', 'MF-BPR', and 'LightGCN-BPR'. To ensure fair comparisons and avoid the impact caused by the L2 regularization term, we removed it from the objective function and fixed the number of epochs to 50. We split items into ten groups based on frequency while keeping the total number of interactions of each group the same, then record the average magnitude of items within each group. Figure 5 shows the results on different datasets. We can observe that:

Table 5. Impact of message passing strategy. 'User' indicates propagating messages on the user side, while 'Item' indicates propagating messages on the item side.

Propagation Side			User			Item		
Setting of $[\alpha_0, \alpha_1]$			[0.5,0]	[0.5,0.5]	[1.0,0]	[0.5,0]	[0.5,0.5]	[1.0,0]
Performance	Gowalla	Recall	0.1156	0.1184	**0.1672**	0.1746	**0.1875***	0.0969
		NDCG	0.0855	0.0828	**0.1373**	0.1461	**0.1567***	0.0656
	Yelp2018	Recall	0.0446	0.0503	**0.0685**	0.0682	**0.0737***	0.0408
		NDCG	0.0349	0.0392	**0.0565**	0.0565	**0.0611***	0.0311
	Amazon-Book	Recall	0.0466	0.0489	**0.0543**	0.0463	**0.0560***	0.0455
		NDCG	0.0366	0.0382	**0.0427**	0.0365	**0.0440***	0.0360
	Alibaba-iFashion	Recall	0.1212	0.1050	**0.1272***	0.1202	**0.1213**	0.0596
		NDCG	0.0592	0.0505	**0.0622***	0.0574	**0.0577**	0.0261

- When using MF as the backbone, the curve of item magnitude learned by SSM within different groups is flat, which confirms that SSM cannot adjust the item magnitude. However, graph-based models (*e.g.,* LightGCN) can adjust item magnitude itself, compensating for the shortcoming of SSM and achieving excellent performance (*cf.* Table 2).
- BPR uses inner product as the similarity function, which allows it to adjust the magnitude of items during training. However, this can lead to popularity bias, especially for graph-based models such as LightGCN-BPR, where popular items typically obtain much larger magnitudes and hence have larger similarity scores. This makes popular items even more popular and results in suboptimal performance.

4.4.6 ***Empirical study on message passing strategy.*** As analyzed in Section 3.2.2, the normalization factors α_0 and α_1, which determine the message passing rule, play important roles in adjusting the magnitude of learned representations. To see how α_0 and α_1 influence the ranking performance, we adopt SVD++ which can be viewed as the simplest message-passing-based method as the backbone model, and train it using SSM. Note that the vanilla SVD++ only propagates messages on the user side, here we introduce a variant that only propagates messages on the item side. Similar to the formulae of vanilla SVD++ on the user side, we define the propagation rule of SVD++ on the item side as follows:

$$\boldsymbol{p}_u = \boldsymbol{p}_u^{(0)}, \quad \boldsymbol{q}_i = \boldsymbol{q}_i^{(0)} + \sum_{u \in \mathcal{P}_i} \frac{1}{|\mathcal{P}_i|^{\alpha_0} |\mathcal{P}_u|^{\alpha_1}} \boldsymbol{p}_u^{(0)}, \tag{30}$$

where $\boldsymbol{p}_u^{(0)}$ (or $\boldsymbol{q}_i^{(0)}$) and $\boldsymbol{p}_u$ (or $\boldsymbol{q}_i$) are the ID embedding and final representation of user u (or item i), respectively. Table 5 shows the Recall@20 and NDCG@20 metrics on all four datasets. We can observe that:

- With different values of α_0 and α_1, the ranking performance varies considerably. Typically, compared with α_1, changing the value of α_0 will cause larger performance fluctuations. This is in line with our analyses that α_0 determines the order of the representation magnitude while α_1 influences its multiplication factor.
- The best choices of α_0 and α_1 are different in user-side SVD++ and item-side SVD++. More specifically, on all datasets, the best performance of user-side SVD++ is achieved when $\alpha_0 = 1.0$ and $\alpha_1 = 0$. While for item-side SVD++, $\alpha_0 = \alpha_1 = 0.5$ is the best option across four datasets.
- On Gowalla, Yelp2018, and Amazon-Book datasets, item-side SVD++ can obtain better performance than user-side SVD++, which is in contrast to the case on Alibaba-iFashion.

Table 6. Efficiency comparison on Amazon-Books using a three-layer LightGCN as backbone model. N is the number of negative samples.

Loss	Time/Epoch	Best Epoch	Total Time
BCE (N=4)	268s	61	4h32m
BPR (N=1)	48s	700	9h20m
BPR (N=2048)	1983s	45	24h47m
SM	81s	29	39m
CCL (N=2048)	124s	34	1h10m
SSM (N=2048)	39s	21	13m

4.4.7 ***Efficiency Comparison.*** We study the training efficiency of SSM. Specifically, we conduct experiments on the same Nvidia Titan RTX graphics card equipped with an Inter i7-9700K CPU (32GB Memory). We compare them under the same implementation framework based on TensorFlow, using the same acceleration methods (*i.e.,* accelerating the sampling with C++) to ensure fairness. The backbone model is a three-layer LightGCN. The results are reported in Table 6. We can find that equipped with in-batch negative sampling and temperature-aware cosine similarity, SSM is much more efficient than other baselines in both the average training time per epoch and the number of epochs to reach the best performance, meanwhile achieving the leading performance (*cf.* Table 2). Surprisingly, SM achieves the second-best training efficiency on the Amazon-Book dataset. However, as mentioned previously, computing SM can be computationally expensive for large datasets since it involves exponentiating all the scores of candidate items. To verify this, we enriched the candidate item pool with a preset number (#Paddings in Table 7) while keeping the training and testing samples unchanged. We then recorded the GPU memory and time cost per epoch of SM and SSM using a three-layer LightGCN as the backbone model on the Amazon-Book dataset, as shown in Table 7. Our results demonstrate that the time cost of SM increases linearly as the number of padding items increases. When the padding number reaches 500,000, SM suffers from out-of-memory (OOM) errors. In contrast, SSM's memory cost is low, and the running time remains almost unchanged across different numbers of padding items. These findings indicate that SSM is more efficient than SM in terms of memory usage and may be a better choice when dealing with real-world large-scale recommender systems.

Table 7. Memory cost comparison on Amazon-Books using a three-layer LightGCN as backbone model.

#Paddings	SM		SSM	
	GPU Memory	Time/Epoch	GPU Memory	Time/Epoch
0	4865M	81s	1793M	39s
100,000	5873M	132s	1793M	41s
200,000	9973M	183s	2801M	42s
300,000	9973M	231s	2801M	43s
400,000	10993M	280s	2801M	44s
500,000	OOM	-	2801M	45s

5 RELATED WORK

In this section, we review the popular objective functions for item recommendation which cast the task into a supervised learning problem.

Many traditional item recommendation methods perform learning by minimizing the **pointwise** divergence of the reconstructed interaction matrix from the observed interaction matrix. According to the way the divergence is defined, pointwise losses can be further categorized into mean square error loss [21, 25, 31, 52], binary cross-entropy loss [19, 40], and hinge loss [38], to name a few. However, for item recommendation, the primary goal is not to score the interaction exactly to 1 or 0. Instead, the relative ordering of a positive item over an unobserved item is of central importance. Toward this goal, **pairwise** losses are proposed to optimize the preference structure consistency between the reconstructed interaction matrix from the observed interaction matrix. Specifically, pairwise losses treat training data as a set of triplet instances $\{(u, i, j)\}$ to capture the intention that user u prefers to positive item i over irrelevant item j. BPR [37] is one of the most popular pairwise losses in item recommendation which is proven to optimize AUC scores. WARP [45] is another pairwise algorithm customized for item recommendation. It encourages the predicted score of positive item larger than that of negative item above a margin, which is in line with the hinge loss. Similarly, [33] proposed a max-margin hinge ranking loss to minimize the ranking risk in the reconstructed matrix. [22] devised a pairwise MSE loss that computes the relative difference between the actual non-zero and zero entries and the difference between their corresponding predicted entries.

Besides pointwise losses and pairwise losses, a third option for item recommendation is to model the recommendation predictions over all items into a probability distribution that is normalized using a softmax function, termed the **softmax** loss [35]. It's worth mentioning that in the community of Learning-to-Ranking, this type of loss function is generally termed list-wise loss [5, 47]. We will not go deep into it since it is not within the scope of this work. Prior work [4] verified that softmax loss aligns well with the ranking metrics. However, calculating the partition function of softmax is computationally costly. A substitute for softmax loss is SSM loss which reduces the computational cost by employing the partition function only on a small subset of negatives. However, common view on SSM loss is that it's a biased version of full softmax loss which can be corrected by log$Q$ correction [2]. Agreeing with this point of view, some follow-on works in recommendation devise different methods to get unbiased estimation[1, 3, 43, 48, 50]. For example, Blanc and Rendle [3] proposed a divide and conquer algorithm to estimate the proposal distribution with kernel based sampling strategy. Yi et al. [50] presented a method to estimate item frequency for streaming data without requiring fixed item vocabulary. Wang et al. [43] devised a cross-batch negative sampling with the FIFO memory bank to improve the accuracy of estimating item distribution by enlarging the number of negative samples.

Most recently, a surge of works introduced contrastive loss like InfoNCE loss into recommendation [30, 41, 46, 55]. At its core is maximizing the agreement of positive pairs as compared with that of negative pairs. Interestingly, when directly utilizing the supervision signal (whether the interaction is observed) of the data to construct positive and negative pairs, InfoNCE loss becomes SSM loss. However, to the best of our knowledge, only very limited works [29, 41, 55] utilized SSM loss as their main task objective to train recommendation model. Zhou et al. [55] proved that both contrastive loss and multinomial IPW loss optimize the same objective in principle but failed to answer why contrastive loss works. Liu et al. [29] presented a debiased contrastive loss to remedy the negative effects of false negative samples in the randomly sampled subset. MSCL [41] is mostly related to our work which also realized the superiority of using SSM loss to train recommender. However, it differs from our work in: (1) MSCL [41] only empirically verified the effectiveness of SSM loss for graph-based recommender. In contrast, we theoretically reveal three advantages of SSM loss, that is, mitigating popularity bias, mining hard negative samples, and maximizing the ranking metrics; (2) We also uncover the shortcomings of SSM loss in adjusting the representation magnitudes and present theoretical evidence to support our argument.

6 CONCLUSION AND FUTURE WORK

In this work, we present insightful analyses of SSM for item recommendation. Firstly, we theoretically disclose model-agnostic advantages of SSM in mitigating popularity bias, mining hard negative samples and maximizing ranking metrics. We then probe the model-specific characteristics of SSM and point out its potential shortcoming in adjusting representation magnitudes. To justify our argument, we further show that message passing based methods are capable of adjusting magnitude. We conducted extensive experiments on four benchmark datasets, demonstrating the superiority of training history- and graph-based models using SSM for both normal and long tail recommendation tasks.

We believe the comprehending of SSM is inspirational to future developments of recommendation community. In future work, we would like to further improve the drawbacks of SSM, making it more powerful for item recommendation. Moreover, since practical recommender systems usually involve rich side information, exploring the potential of SSM for feature-based CTR models is another promising direction.

REFERENCES

[1] Yu Bai, Sally Goldman, and Li Zhang. 2017. TAPAS: Two-pass Approximate Adaptive Sampling for Softmax. *CoRR* abs/1707.03073 (2017).

[2] Yoshua Bengio and Jean-Sébastien Senecal. 2003. Quick Training of Probabilistic Neural Nets by Importance Sampling. In *AISTATS*.

[3] Guy Blanc and Steffen Rendle. 2018. Adaptive Sampled Softmax with Kernel Based Sampling. In *ICML*. 589–598.

[4] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *ICTIR*. 75–78.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*. 129–136.

[6] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and Debias in Recommender System: A Survey and Future Directions. *ACM Trans. Inf. Syst.* 41, 3 (2023), 67:1–67:39.

[7] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*. AAAI Press, 27–34.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*. 1597–1607.

[9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. 191–198.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[11] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP*. 6894–6910.

[12] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised Representation Learning by Predicting Image Rotations. In *ICLR*.

[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, Vol. 9. 249–256.

[14] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming Session-based Recommendation. In *KDD*. 1569–1577.

[15] Michael Gutmann and Aapo Hyvärinen. 2012. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *J. Mach. Learn. Res.* 13 (2012), 307–361.

[16] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.

[17] 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.

[18] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *TKDE* 30, 12 (2018), 2354–2366.

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[20] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.
[21] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. 263–272.
[22] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *SIGKDD*. 659–667.
[23] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. In *NeurIPS*.
[24] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[25] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*. 426–434.
[26] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
[27] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.
[28] Defu Lian, Qi Liu, and Enhong Chen. 2020. Personalized Ranking with Importance Sampling. In *WWW*. 1093–1103.
[29] Zhuang Liu, Yunpu Ma, Yuanxin Ouyang, and Zhang Xiong. 2021. Contrastive Learning for Recommender System. *CoRR* abs/2101.01317 (2021).
[30] 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*.
[31] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *ICDM*. 497–506.
[32] Vilfredo Pareto. 1964. *Cours d'économie politique*. Vol. 1. Librairie Droz.
[33] Dohyung Park, Joe Neeman, Jin Zhang, Sujay Sanghavi, and Inderjit S. Dhillon. 2015. Preference Completion: Large-scale Collaborative Ranking from Pairwise Comparisons. In *ICML*. 1907–1916.
[34] Ankit Singh Rawat, Jiecao Chen, Felix X. Yu, Ananda Theertha Suresh, and Sanjiv Kumar. 2019. Sampled Softmax with Random Fourier Features. In *NeurIPS*. 13834–13844.
[35] Steffen Rendle. 2022. Item Recommendation from Implicit Feedback. In *Recommender Systems Handbook*. Springer US, 143–171.
[36] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*. 273–282.
[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.
[38] Jason D. M. Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*. 713–719.
[39] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *ICML*. 1670–1679.
[40] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In *SIGKDD*. 255–262.
[41] Hao Tang, Guoshuai Zhao, Yuxia Wu, and Xueming Qian. 2023. Multisample-Based Contrastive Loss for Top-K Recommendation. *IEEE Trans. Multimedia* 25 (2023), 339–351.
[42] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018). http://arxiv.org/abs/1807.03748
[43] Jinpeng Wang, Jieming Zhu, and Xiuqiang He. 2021. Cross-Batch Negative Sampling for Training Two-Tower Recommenders. In *SIGIR*. 1632–1636.
[44] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.
[45] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: Scaling Up to Large Vocabulary Image Annotation. In *IJCAI*. 2764–2770.
[46] 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.
[47] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *ICML*. 1192–1199.
[48] 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 *Companion of The Web Conference*. 441–447.
[49] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix X. Yu, Ting Chen, Aditya Krishna Menon, Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi (Jay) Kang, and Evan Ettinger. 2021. Self-supervised Learning for Large-scale Item

Recommendations. In *CIKM*. 4321–4330.
[50] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed H. Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *RecSys*. 269–277.
[51] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*. 974–983.
[52] Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, S. V. N. Vishwanathan, and Inderjit Dhillon. 2014. NOMAD: Non-Locking, Stochastic Multi-Machine Algorithm for Asynchronous and Decentralized Matrix Completion. 7, 11 (2014), 975–986.
[53] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR*. 785–788.
[54] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *SIGIR*. 11–20.
[55] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems. In *SIGKDD*. 3985–3995.
[56] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. In *CIKM*. 1893–1902.