Personalized Ranking with Importance Sampling

Defu Lian, Qi Liu, Enhong Chen
Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology &
School of Data Science, University of Science and Technology of China
{liandefu,qiliuql,cheneh}@ustc.edu.cn

ABSTRACT

As the task of predicting a personalized ranking on a set of items, item recommendation has become an important way to address information overload. Optimizing ranking loss aligns better with the ultimate goal of item recommendation, so many ranking-based methods were proposed for item recommendation, such as collaborative filtering with Bayesian Personalized Ranking (BPR) loss, and Weighted Approximate-Rank Pairwise (WARP) loss. However, the ranking-based methods can not consistently beat regression-based models with the gravity regularizer. The key challenge in ranking-based optimization is difficult to fully use the limited number of negative samples, particularly when they are not so informative. To this end, we propose a new ranking loss based on importance sampling so that more informative negative samples can be better used. We then design a series of negative samplers from simple to complex, whose informativeness of negative samples is from less to more. With these samplers, the loss function is easy to use and can be optimized by popular solvers. The proposed algorithms are evaluated with five real-world datasets of varying size and difficulty. The results show that they consistently outperform the state-of-the-art item recommendation algorithms, and the relative improvements with respect to NDCG@50 are more than 19.2% on average. Moreover, the loss function is verified to make better use of negative samples and to require fewer negative samples when they are more informative.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**.

KEYWORDS

Personalized Ranking, Negative Sampling, Implicit Feedback

ACM Reference Format:
Defu Lian, Qi Liu, Enhong Chen. 2020. Personalized Ranking with Importance Sampling. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380187

1 INTRODUCTION

With rapid development of IT technology, information in the network grows explosively. When making decisions in daily life, people often have to access large mounts of information. For example, there are tens of thousands of movies in Netflix, millions of books in Douban, and billions of webpage collection in Del.icio.us. Therefore, people encounter the problem of information overload. Recommendation techniques are an important way to address information overload by filtering unintended information. They have been evolving for more than 30 years and are widely used in E-commerces, advertisements and many other scenarios. Many companies, such as Amazon and Taobao, benefit a lot from recommendation techniques. There are two tasks in recommendation – rating prediction and item recommendation. Rating prediction is to estimate unseen ratings based on user rating history while item recommendation is to predict a personalized ranking on a set of items. Intuitively, item recommendation is more directly correlated with information filtering, and has become a central research topic, particular with the advancement of deep recommendation models.

Item recommendation is commonly investigated for implicit feedback, and faces with the one-class problem, where negative and unlabeled positive examples are mixed together in unobserved data. To address the one-class problem, two representative types of algorithms are proposed for item recommendation from implicit feedback. The one type is regression-based models, including Weighted Regularized Matrix Factorization [9, 12, 18, 26] (WRMF) and Sampling-based wALS [27]. The former idea is to treat all unobserved entries as negative but to assign them lower weights than positive samples. It has been proved that WRMF intrinsically imposes the gravity (or implicit) regularizer, which penalizes the non-zero predictions of all unobserved values [1, 14]. The latter idea is to draw an approximately same size of negative examples to positive ones, and also assign them lower weights. The lower weight corresponds to lower confidence of unobserved entries being negative. The systematic study in [37] reveals that treating all unobserved entries as negative performs better than subsampling from them.

The other type is ranking-based models, including collaborative filtering with Bayesian Personalized Ranking (BPR) loss [29] and Weighted Approximate-Rank Pairwise (WARP) loss [15, 36]. BPR optimizes AUC while WARP optimizes precision. Since AUC penalizes inconsistent pairs equally, BPR is usually inferior to WARP. Moreover, the sampler of WARP always draws informative negative samples, the score of which should be larger than the positive example minus one. The sampler is more effective than uniform sampling in BPR, but much less efficient due to scanning more and more candidates along the training course. The sampler in BPR was also improved by oversampling informative pairs to speed up convergence and improve accuracy [28].

Though optimizing ranking loss aligns better with the ultimate goal of item recommendation, the ranking-based methods can not consistently beat the regression-based models with the gravity regularizer. The main reason is that the samplers usually draw a limited number of negative samples for the sake of efficiency

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380187

while more negative samples usually lead to better performance. Therefore, the key challenge in the ranking-based optimization is how to make better use of the limited number of negative samples. To this end, we propose a new Personalized Ranking loss based on Importance Sampling, PRIS for short, so that more informative negative samples are assigned larger weights/importances. Since more informative negative samples can contribute more to gradient of the ranking loss function, the magnitude of gradient becomes much larger so that training can be accelerated. We observe that even using the uniform sampler, the model with the new ranking loss can perform much better than BPR, as shown in Figure 3. Following that, we develop a series of negative samplers from simple to complex. The informativeness of negative samples drawn from these samplers is from less to more. Whichever sampler is used, the new loss function can be similarly optimized by any popular solvers, such as SGD and ADAM.

The contributions in this paper are summarized as follows:

- We propose a new Personalized Ranking loss based on Importance Sampling, PRIS for short, which can assign larger weights to more informative negative samples. This can ensure larger magnitude of gradient along the training course, so as to speed up convergence. The advantage of the new loss function can be easily observed in Figure 3, by comparing PRIS of the uniform sampler (PRIS-Uniform) with BPR.
- We develop four negatives samplers from simple to complex based on rejection sampling, and for the first time systematically investigate efficiency of negative sampling, reporting average sampling times to draw a negative sample. Two complex negative samplers need to learn their own parameters, incurring extra time overhead to sampling. To reduce time cost, we propose a joint learning framework by simultaneously learning model parameters and sampler parameters.
- We evaluate the proposed algorithms with five real-world datasets of varying size and difficulty. The experimental results show that the proposed algorithms can consistently outperform the state-of-the-art item recommendation algorithms, including BPR with adaptive oversampling, WRMF and WARP. On average, the relative improvements with respect to NDCG@50 are over 19.2%. Furthermore, when negative samples are more informative, the model performs better and requires the fewer samples.

2 RELATED WORK

We first review recent advance of item recommendation from implicit feedback, and then survey recent important techniques for negative sampling in deep learning community.

2.1 Item Recommendation from Implicit Feedback

Item recommendation from implicit feedback dates back to about 10 years ago, when WRMF [12], OCCF [27] and BPR [29] were invented. Since truly negative are mixed together with unlabeled positive, these algorithms differ from each other in how to use unobserved values and whether to use ranking loss. WRMF treated all unobserved entries as negative but assigned them lower confidence of being negative. Instead of using all unobserved values, OCCF only subsampled some from them. Later, OCCF was improved in [26] by also considering all unobserved values. These algorithms used weighted regression loss for parameter learning. They were systemically compared in [37], concluding that treating all unobserved entries as negative performed better. Following that, an implicit/gravity regularizer was decomposed in the loss function of WRMF according to [1] and revealed to play an important role in item recommendation from implicit feedback. Then the regularizer was imposed to the item recommendation models with side information [14, 18, 19]. Note that the regurlarizer penalizes non-zero predictions of unobserved values via a quadratic function, so that parameter learning can be very efficient.

Similar to OCCF, BPR also subsampled unobserved entries as negative via uniform sampling, but utilized the sigmoid-based ranking loss for parameter learning. By observing small gradient magnitude resulting from uniform sampling, adaptive and context-dependent sampling was proposed in [28] to speed up convergence and improve accuracy. Due to the non-linear loss, BPR can not efficiently learn the parameters when all unobserved entries are considered negative. The situation can be changed in RankALS [31] by replacing the sigmoid-based ranking loss with the quadratic ranking loss. RankALS was further improved by imposing a ranking-based implicit regularizer [3], by penalizing the large difference of prediction between any two unobserved values. Since BPR optimized AUC, which treats all inconsistent pairs equally, BPR may not be best suitable for the top-k item recommendation. Therefore, the Weighted Approximate-Rank Pairwise (WARP) loss was proposed to optimize the precision [15, 36]. WARP also used uniform sampling with rejection to draw more informative negative samples, the score of which should be larger than that of the positive example minus one. WARP was also used for collaborative metric learning [11], leading to the state-of-the-art item recommendation method.

2.2 Techniques of Negative Sampling

In many natural language applications, it is very computational expensive to represent an output distribution over the choice of a word based on the softmax function. To address the efficiency, many approximate algorithms were proposed. For example, hierarchical softmax [24] and lightRNN [16] decomposed the probabilities, and Contrastive Divergence [10] approximated the gradient based on MCMC. As an alternative, negative sampling is also widely used in reducing the computational cost of training the models. For example, Noise-Contrastive Estimation [8] performed non-linear logistic regression to discriminate between the observed data and some artificially generated noise, and was successfully applied for language modeling by considering the unigram distribution of training data as the noise distribution [23]. The noise distribution plays an important role for approximation accuracy, and many recent work were proposed for improving the noise distribution. Generative Adversarial Networks [5, 34] directly learned the noise distribution via the generator networks. Self-Contrast Estimator [6] copied the model and used it as the noise distribution after every step of learning. However, when applied for recommendation algorithms, these two methods suffer from high computational cost for sampling from the noise distribution. To improve sampling efficiency, the self-adversarial negative sampling [30] draws negative samples

from the uniform distribution but treats the sampling probability as their weights. Dynamic negative sampling[38] also draws a set of negative samples from the uniform distribution and then picks the item with the largest prediction score. Kernel-based sampling [2] drew samples proportionally to a quadratic kernel, making it fast to compute the partition function in the kernel space and to sample entries in a divide and conquer way.

3 PRELIMINARIES

Let $\mathcal{U}$ be the set of M users and $\mathcal{I} = \{o_1, \cdots, o_N\}$ the set of N items. The model is assumed to train from user-item interaction history, i.e., $\mathcal{S} = \{(u, i) \in \mathcal{U} \times \mathcal{V} | \text{user u interacts with item i}\}$. Let $\mathcal{I}_u$ denote positive items with which user u interacts, and $\mathcal{U}_i$ denote the users who interact with the item i.

3.1 Bayesian Personalized Ranking

BPR is a pairwise ranking optimization framework for one-class collaborative filtering. The objective function of BPR is formulated as follows:

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{S}} \sum_{j\notin\mathcal{I}_u} \ln \sigma(\hat{x}_{uij}) + \lambda\|\Theta\|^2, \tag{1}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function and $\lambda\|\Theta\|^2$ is a regularization term. $\sigma(\hat{x}_{uij})$ is modeled as the probability that a user u prefers item i to item j. Minimizing $\mathcal{L}$ makes the score of positive items larger than negative items. Note that $\ln \sigma(x)$ is a differentiable surrogate function of the Heaviside function, so BPR approximately optimizes the ranking statistic AUC (Area under the ROC curve). Since $\mathcal{I}\backslash\mathcal{I}_u$ is overly large, it is not efficient to cycle through all of them. Therefore, learning BPR is typically based on Stochastic Gradient Descent (SGD), which uniformly samples an item j from the set and performs an update for the triple. However, such SGD learning algorithms are very slow to converge, particularly when item popularity has a tailed distribution [28]. This is because when BPR is better trained, positive item i can be easily distinguished from a uniformly-sampled item j (i.e., $\sigma(\hat{x}_{uij}) \to 1$), so that the triple contributes little to gradient, i.e., $(1-\sigma(\hat{x}_{uij}))\frac{\partial \hat{x}_{uij}}{\partial \Theta} \to 0$. Given the gradient of small magnitude, the overall update to the parameter Θ is very small. To address this issue, several advanced samplers were proposed in [28] for speeding up convergence. However, the recommendation quality is not improved a lot by user-dependent (or context-dependent) samplers, and the static samplers even lead to performance degradation.

4 PERSONALIZED RANKING VIA IMPORTANCE SAMPLING

In this paper, we propose a new framework for personalized ranking, to address the problem in BPR. Direct optimization in the framework is very time-consuming, so we derive a new loss function based on importance sampling. Since the proposal distribution is very important for importance sampling, we introduce four negative samplers with varying complexity, and then study sampling efficiency and learning efficiency of samplers. We finally propose a joint learning framework for both the model and samplers, to reduce extra overhead of learning sampler parameters.

4.1 The New Loss Function

One important problem in BPR lies in ignorance of negative confidence, so that all negative items are treated equally. In the new framework, we first model the confidence as a probability of items being negative. Recall that $\sigma(\hat{x}_{uij})$ denotes the probability that a user u prefers item i to item j. Thus a smaller $\hat{x}_{uij}$ corresponds to lower likelihood that the user u prefers item i to item j, implying higher probability of item j being negative. Therefore, in this paper, we model the negative probability of unobserved items as follows:

$$P(j|u,i) = \frac{\exp(-\hat{x}_{uij})}{\sum_{j'\in\mathcal{I}\backslash\mathcal{I}_u}\exp(-\hat{x}_{uij'})}. \tag{2}$$

By considering this probability as the confidence/weight of negative item j, we formulate the new framework in the following way.

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{S}} \sum_{j\in\mathcal{I}\backslash\mathcal{I}_u} P(j|u,i)\ln\sigma(\hat{x}_{uij}) + \lambda\|\Theta\|^2. \tag{3}$$

Therefore, the framework pays more attention to unobserved items with higher negative probability. Due to a large size of unobserved items, it is not efficient to cycle through all of them to perform a update. Because of high-cost computation of negative probability, it is also not efficient to approximate the loss by directly sampling a fixed size of items from negative probability. Instead, by considering the second summation $\mathcal{L}(u,i) = \sum_{j\in\mathcal{I}\backslash\mathcal{I}_u} P(j|u,i)\ln\sigma(\hat{x}_{uij})$ as expectation with respect to the negative probability, we derive the loss approximation based on importance sampling. In particular, assuming that the proposal distribution $Q(j|u,i)$ is easy to sample and L items are sampled from this distribution, the second summation $\mathcal{L}(u,i)$ is approximated by

$$\mathcal{L}(u,i) \approx \frac{1}{L}\sum_{l=1}^{L}\frac{P(j_l|u,i)}{Q(j_l|u,i)}\ln\sigma(\hat{x}_{uij_l}),\ j_l \sim Q(j|u,i). \tag{4}$$

Since it is also time-consuming to compute the normalization constant of $P(j|u,i)$, it should also be approximated by the L samples. In particular,

$$Z_P = \sum_{j\in\mathcal{I}\backslash\mathcal{I}_u}\exp(-\hat{x}_{uij}) \approx Z_Q\frac{1}{L}\sum_{l=1}^{L}\exp(-\hat{x}_{uij_l} - \ln\tilde{Q}(j_l|u,i)) \tag{5}$$

where $\tilde{Q}(j_l|u,i)$ is the unnormalized probability of $Q(j_l|u,i)$, such that $Z_Q = \sum_{j\in\mathbb{I}\backslash\mathbb{I}_u}\tilde{Q}(j_l|u,i)$ is a normalization constant. Letting $w_l = \frac{\exp(-\hat{x}_{uij_l} - \ln\tilde{Q}(j_l|u,i))}{\sum_k \exp(-\hat{x}_{uij_k} - \ln\tilde{Q}(j_k|u,i))}$ denote the weight of sample j_l, the approximate loss is then formulated as follows:

$$\mathcal{L} \approx -\sum_{(u,i)\in\mathcal{S}}\sum_{l=1}^{L} w_l \ln\sigma(\hat{x}_{uij_l}) + \lambda\|\Theta\|^2,\ j_l \sim Q(j|u,i). \tag{6}$$

If the proposal distribution Q is the exact P, $w_l = \frac{1}{L}$ and the approximation is reduced back to the BPR loss. Otherwise, the sample weight can compensate the deviation between P and Q to some extent. Therefore, even static samplers can be likely to speed up convergence and improve the recommendation quality.

4.2 Negative Samplers

The new framework for personalized ranking is approximated by importance sampling. Given the fixed number of samples, the quality of negative samplers determines approximation accuracy. In this part, we investigate the design of negative samplers and study sampling efficiency. Before discussing complex samplers, we first revisit the static samplers and specialize the sample weights.

Below we consider the simplest case of modeling $\hat{x}_{uij}$, i.e., $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$, where $\hat{x}_{ui}$ denotes prediction score of user u for item i. Then the negative probability of a unobserved item j is modeled as $P(j|u,i) = \frac{\exp \hat{x}_{uj}}{\sum_{j' \in \mathcal{I} \setminus \mathcal{I}_u} \exp(\hat{x}_{uj'})}$, which is independent of positive item i. In other words, unobserved items with higher prediction scores are more likely to be negative. Note that unobserved items with the top-k highest scores are also recommended to users, any algorithms which resort to deterministic selection rather than stochastic sampling may suffer from the false negative issue. The approximation in stochastic sampling further alleviates the issue according to Table 3.

4.2.1 Uniform Sampling. Uniform sampling over unobserved items is based on the simplest uniform proposal, which is defined as

$$Q^U(j|u,i) = \begin{cases} 0, & \text{if } j \in \mathcal{I}_u \\ \frac{1}{N - N_u}, & \text{otherwise} \end{cases} \tag{7}$$

where $N_u = |\mathcal{I}_u|$ denotes the number of items with which user u interacts. Directly sampling items from this proposal is easy but not memory-efficient or computation-efficient, due to the maintenance of the set $\mathcal{I} \setminus \mathcal{I}_u$ for each user. Therefore, Algorithm 1 is usually used in practice. We will show that the algorithm is equivalent to rejection sampling [25], where the proposal distribution is a uniform distribution over all items. The weight for sample j_l in Eq (6) is then defined as

$$w_l^U = \frac{\exp(\hat{x}_{uj_l})}{\sum_{t=1}^{L} \exp(\hat{x}_{uj_t})}. \tag{8}$$

Algorithm 1: Uniform Sampling

Input: $\mathcal{I}_u$ – positive items of user u
Output: $\mathcal{E}_u$ – list of negative items

$\mathcal{E}_u \leftarrow \emptyset$;
for $l \leftarrow 1$ **to** L **do**
  repeat
    $n \leftarrow$ randint$(1, N)$; `// integer from 1 to N`
    $j \leftarrow o_n$;
  until $j \in \mathcal{I} \setminus \mathcal{I}_u$;
  insert$(\mathcal{E}_u, j)$

4.2.2 Popularity Sampling. If a user does not interact with a popular item, she may be truly uninterested in it since the item is highly likely to be exposed to the user. When recommending items based on popularity, it was reported that it performed much better than random-based recommendation. Therefore, sampling from popularity distribution should be beneficial. The popularity proposal is defined as

$$Q^P(j|u,i) = \begin{cases} 0, & \text{if } j \in \mathcal{I}_u \\ \frac{f_j}{\sum_{j' \in \mathcal{I} \setminus \mathcal{I}_u} f_{j'}}, & \text{otherwise.} \end{cases} \tag{9}$$

where f_i is (normalized) popularity of item i. Let c_i be occurring frequency of item i, f_i can be set to c_i, $\log(1 + c_i)$ and $c_i^{3/4}$[22]. It is also time-consuming to directly sample items from this proposal and memory-consuming to store the proposal in advance. Therefore, Algorithm 2 is also based on rejection sampling, whose proposal distribution is popularity distribution over all items, i.e., $P_Q(i) = f_i / \sum_i f_i$. The weight for a sample j_l is defined as

$$w_l^P = \frac{\exp(\hat{x}_{uj_l} - \ln f_{j_l})}{\sum_{t=1}^{L} \exp(\hat{x}_{uj_k} - \ln f_{j_t})}. \tag{10}$$

Algorithm 2: Popularity Sampling

Input: $\mathcal{I}_u$, cumulative probability $C_Q[i] = \sum_{j=1}^{i} P_Q(i)$
Output: $\mathcal{E}_u$ – list of negative items

$\mathcal{E}_u \leftarrow \emptyset$;
for $l \leftarrow 1$ **to** L **do**
  repeat
    $r \leftarrow$ rand() ; `// uniform dist. over [0, 1)`
    $n \leftarrow$ bisect(C_Q, r); `// binary search with r, such that` $C_Q[n+1] > r$ `and` $C_Q[n] <= r$
    $j \leftarrow o_n$;
  until $j \in \mathcal{I} \setminus \mathcal{I}_u$;
  insert$(\mathcal{E}_u, j)$

Before introducing next samplers, we try to generalize from the above samplers. In particular, first defining $P_Q(j|u,i)$ as an upper envelope of the unnormalized version $\tilde{Q}(j|u,i)$, we can then define $Q(j|u,i)$ as follows:

$$Q(j|u,i) = \begin{cases} 0, & \text{if } j \in \mathbb{I}_u \\ \frac{P_Q(j|u,i)}{\sum_{j' \in \mathcal{I} \setminus \mathcal{I}_u} P_Q(j'|u,i)}, & \text{otherwise} \end{cases} \tag{11}$$

In order to efficiently sample items from $Q(j|u,i)$, we can apply the following theorem.

Theorem 4.1. *Sampling from $Q(j|u,i)$ in Eq (11) is equivalent to rejection sampling with the proposal distribution $P_Q(j|u,i)$.*

Proof. According to rejection sampling, we first sample $j \sim P_Q(j|u,i)$ and then sample $u \sim \mathrm{U}(0,1)$. If $u > \frac{\tilde{Q}(j|u,i)}{P_Q(j|u,i)}$, the sample is rejected, otherwise it is accepted. If $j \in \mathcal{I}_u$, $\tilde{Q}(j|u,i) = 0$ so that the sample is rejected with probability one; otherwise, $\tilde{Q}(j|u,i) = P_Q(j|u,i)$ so that it is accepted with probability one. We then follow the rejection sampling theorem [25] for completing proof. □

Based on the theorem, we can derive the probability of acceptance and expected times of sampling.

Corollary 4.2. *The probability of acceptance is $P(accept) = 1 - P_Q(\mathcal{I}_u|u,i)$, where $P_Q(\mathcal{I}_u|u,i) = \sum_{j \in \mathcal{I}_u} P_Q(j|u,i)$.*

Corollary 4.3. *Let X_u be a random variable of sampling times in rejection sampling. Then X_u follows the geometric distribution, s.t. $P(X_u = n) = P(accept)(1 - P(accept))^{n-1}$, and $E[X_u] = \frac{1}{P(accept)}$.*

Based on this corollary, we can deduce that the time complexity of uniform sampling is $O(\frac{LN}{N - N_u})$. Thus uniform sampling is very efficient. The time complexity of popularity sampling is $O(\frac{L \log(N)}{1 - P_Q(\mathcal{I}_u)})$, highly depending on the popularity of user's positive items.

4.2.3 Cluster-Uniform Sampling. The aforementioned samplers are static, user-independent, so the probability of each unobserved item being negative is the same with respect to all users. Note that importance sampling depends crucially how well the proposal distribution matches the desired distribution. To obtain the proposal distribution closer to $P(j|u,i)$, we cluster items into several groups based on item representation in the most recent model, such that each group of unobserved items shares the same probability of being negative. Consequently, the sampling procedure is decomposed into two steps. The first is to sample a group and the second is to sample an item within a group. Let $\boldsymbol{v}_i$ be representation vector of item i and $\boldsymbol{V}$ a matrix form of representation of all items. These items are clustered into K groups based on K-means clustering. The k-th group is represented by a center $\boldsymbol{c}_k$. Let $\boldsymbol{y}$ denote a cluster label vector, such that $y_i \in \{1, \cdots, K\}$ indicates which group the item i belongs to. Let $\mathcal{I}^k$ denote items belonging to the k-th group. Each $\boldsymbol{v}_i$ is then approximated by $\boldsymbol{c}_{y_i}$. Subsequently, the proposal distribution is defined as

$$Q^{CU}(j|u,i) = \begin{cases} 0, & \text{if } j \in \mathcal{I}_u \\ \frac{\exp(\hat{x}_{uy_j})}{\sum_{j' \in \mathcal{I} \setminus \mathcal{I}_u} \exp(\hat{x}_{uy_{j'}})}, & \text{otherwise.} \end{cases} \tag{12}$$

where $\hat{x}_{uy_j}$ predicts the preference score of user u for the group to which item j belongs. The weight for a sample j_l is

$$w_l^{CU} = \frac{\exp(\hat{x}_{uj_l} - \hat{x}_{uy_{j_l}})}{\sum_{t=1}^{L} \exp(\hat{x}_{uj_t} - \hat{x}_{uy_{j_t}})}. \tag{13}$$

There are two ways to sample from this proposal distribution. The one directly follows Theorem 4.1, but once the sample is rejected, the method requires to re-sample a group. This may lead to computational issue, since sampling a group costs more than sampling an item within the group. The other is based on the following theorem, which places rejection sampling into the second step. The procedure is shown in Algorithm 3, where each of L samples only requires to sample one group. Let X_u^{CU} be a random variable of sampling times, then the mean sampling times $E[X_u^{CU}] = \frac{\sum_k N^k \exp(\hat{x}_{uk})}{\sum_k (N^k - N_u^k)\exp(\hat{x}_{uk})}$. The time complexity of Algorithm 3 is $O(E[X_u^{CU}]L + L\log K + KT)$, where T is the time cost of the predict function. Here, we do not take the cost of clustering into consideration.

Theorem 4.4. *Sampling from $Q^{CU}(j|u,i)$ is equivalent to first sampling a group k with probability $P(k|u,i) = \frac{(N^k - N_u^k)\exp(\hat{x}_{uk})}{\sum_{k'}(N^{k'} - N_u^{k'})\exp(\hat{x}_{uk'})}$ and then uniformly sample an item within the group k with rejection. Here N_k is the number of items in the group k and N_k^u is the number of positive items of user u within the group k.*

Proof. $\frac{\exp(\hat{x}_{uy_j})}{\sum_{j' \in \mathcal{I} \setminus \mathcal{I}_u} \exp(\hat{x}_{uy_{j'}})} = \frac{(N^{y_i} - N_u^{y_i})\exp(\hat{x}_{uy_j})}{\sum_k (N^k - N_u^k)\exp(\hat{x}_{uk})} \frac{1}{N^{y_i} - N_u^{y_i}}$. The first part corresponds to the probability $P(y_j|u,i)$. The second part is the probability $P(j|u,y_j)$ to uniformly sample the item j from all unobserved items within the group y_j, which can be efficiently implemented via rejection sampling. □

4.2.4 Cluster-Popularity Sampling. As just discussed, uniform sampling within groups can be improved by popularity sampling, which leads to cluster-popularity sampling. In particular, we define the proposal distribution of cluster-popularity sampling as follows:

$$Q^{CP}(j|u,i) = \begin{cases} 0, & \text{if } j \in \mathbb{I}_u \\ \frac{\exp(\hat{x}_{uy_j} + \ln P_Q(j))}{\sum_{j' \in \mathcal{I} \setminus \mathcal{I}_u} \exp(\hat{x}_{uy_{j'}} + \ln P_Q(j'))}, & \text{otherwise} \end{cases} \tag{14}$$

Similar to Theorem 4.4, we can easily prove Theorem 4.5 to efficiently sample items from this distribution. Algorithm 4 shows the overall procedure. Let X_u^{CP} be a random variable of sampling times, $E[X_u^{CP}] = \frac{\sum_k P_Q(\mathcal{I}^k)\exp(\hat{x}_{uk})}{\sum_k (P_Q(\mathcal{I}^k) - P_Q(\mathcal{I}_u^k))\exp(\hat{x}_{uk})}$. The time complexity of sampling is $O(E[X_u^{CP}]L\log(N/K) + L\log K + KT)$, where we assume the number of items in each group approximately equals N/K. The weight of sample j_l is

$$w_l^{CP} = \frac{\exp(\hat{x}_{uj_l} - \hat{x}_{uy_l} - \ln f_{j_l})}{\sum_{t=1}^{L} \exp(\hat{x}_{uj_t} - \hat{x}_{uy_{j_t}} - \ln f_{j_t})}. \tag{15}$$

Theorem 4.5. *Sampling from $Q^{CP}(j|u,i)$ is equivalent to first sampling a group k with probability $P(k|u,i) = \frac{P_Q(\mathcal{I}^k \setminus \mathcal{I}_u^k)\exp(\hat{x}_{uk})}{\sum_{k'} P_Q(\mathcal{I}^{k'} \setminus \mathcal{I}_u^{k'})\exp(\hat{x}_{uk'})}$ and then sample an item j within the group k with probability $P(j|u,k) = \delta(y_j = k)\frac{P_Q(j)}{P_Q(\mathcal{I}^k \setminus \mathcal{I}_u^k)}$, where $P_Q(\mathcal{I}^k \setminus \mathcal{I}_u^k) = \sum_{j \in \mathcal{I}^k \setminus \mathcal{I}_u^k} P_Q(j) = P_Q(\mathcal{I}^k) - P_Q(\mathcal{I}_u^k)$.*

Algorithm 3: Cluster-Uniform Sampling

Input: $\mathcal{I}_u$, and clustering centers and labels
Output: $\mathcal{E}_u$ – list of negative items

$\mathcal{E}_u \leftarrow \emptyset$;
for $k \leftarrow 1$ **to** K **do**
  $score \leftarrow \text{predict}(u,k)$;
  $prob[k] \leftarrow (N^k - N_u^k)\exp(score)$;
$prob \leftarrow \text{normalize}(prob)$;
$cumprob \leftarrow \text{cdf}(prob)$; // Cumulative probability
for $l \leftarrow 1$ **to** L **do**
  $k \leftarrow \text{bisect}(cumprob, rand())$;
  repeat
    $n \leftarrow \text{randint}(1, N^k - N_u^k)$;
    $j \leftarrow o_n^k$; // The n-th item in group k
  until $j \in \mathcal{I}^k \setminus \mathcal{I}_u^k$;// $\mathcal{I}_u^k = \mathcal{I}_u \cap \mathcal{I}^k$
  $\text{insert}(\mathbb{E}_u, j)$

4.3 A Joint Framework

Although cluster-based methods can sample more informative items from unobserved ones, they require clustering items prior to negative sampling. The overhead can not be ignored particularly when the number of items is large. Moreover, the preference score $\hat{x}_{ui}$ is often modeled by inner product or MLP with user representation and item representation as inputs. Therefore, such score functions may be not a valid metric, due to violation of minimality, triangle inequality and non-negativity. This may lead to clustering items with similar preference scores into different groups [11]. To this end, we propose a joint framework to simultaneously learn model

Algorithm 4: Cluster-Popularity Sampling

Input: $\mathcal{I}_u$, clustering centers and labels, item popularity
Output: $\mathcal{E}_u$ – list of negative items
// user-independent part, to be precomputed
for $k \leftarrow 1$ **to** K **do**
  $P_Q(\mathcal{I}^k) \leftarrow \sum_{j \in \mathcal{I}^k} P_Q(j)$;
  foreach $j \in \mathcal{I}^k$ **do**
    $P_Q^k[j] \leftarrow P_Q(j)/P_Q(\mathcal{I}^k)$;
  $C_Q^k \leftarrow \text{cdf}(P_Q^k)$;
$\mathcal{E}_u \leftarrow \emptyset$; // below are user-dependent
for $k \leftarrow 1$ **to** K **do**
  $P_Q(\mathcal{I}_u^k) \leftarrow \sum_{j \in \mathcal{I}_u^k} P_Q(j)$;
  $score \leftarrow \text{predict}(u, k)$;
  $prob[k] \leftarrow (P_Q(\mathcal{I}^k) - P_Q(\mathcal{I}_u^k)) \exp(score)$;
$prob \leftarrow \text{normalize}(prob)$;
$cumprob \leftarrow \text{cdf}(prob)$;
for $l \leftarrow 1$ **to** L **do**
  $k \leftarrow \text{bisect}(cumprob, rand())$;
  repeat
    $n \leftarrow \text{bisect}(C_Q^k, rand())$;
    $j \leftarrow o_n^k$; // The n-th item in group k
  until $j \in \mathcal{I}^k \backslash \mathcal{I}_u^k$;
  insert($\mathcal{E}_u, j$)

parameters and clustering parameters. The overall framework is illustrated in Figure 1, where three negative items are used and each item are clustered into one of three groups.

The key part is how to formulate clustering so that items with similar preference scores are clustered into the same group. To be generic, we assume similarity between item representation $\boldsymbol{v}_i$ and cluster center $\boldsymbol{c}_k$ is measured by $\text{sim}(\boldsymbol{v}_i, \boldsymbol{c}_k) = \frac{1}{2}\boldsymbol{v}_i^T \boldsymbol{W} \boldsymbol{c}_k + \boldsymbol{w}_I^T \boldsymbol{v}_i + \boldsymbol{w}_C^T \boldsymbol{c}_k$. Here we do not use Euclidean distance for measuring similarity, since it may not the true measurement or even the similarity is not based on a valid metric at all. We let the algorithm directly learn the similarity function. Actually, this similarity function is closely related to several well-known metrics. Given the similarity function, the cluster assignment of item i is achieved by assigning it to the most similar cluster center,

$$y_i = \arg\max_k \text{sim}(\boldsymbol{v}_i, \boldsymbol{c}_k). \tag{16}$$

Then $\boldsymbol{c}_{y_i}$ is an approximation of $\boldsymbol{v}_i$ and can be fed into preference modeling together with user representation to predict an approximate preference score $\hat{r}_{uy_i}$. The preference scores for the positive item and negative items are then fed into the loss function in Eq (6), forming the loss $\mathcal{L}^C$ for learning cluster parameters. Finally, $\mathcal{L}^C + \mathcal{L}$ is optimized for joint learning.

However, the max operator is non-differentiable so that the back-propagation algorithm cannot be applied for computing gradients. Below we derive the continuous relaxation to approximate the gradient of the argmax operator. In particular, by representing each cluster index with an one-hot vector, i.e., $\boldsymbol{b}_i = \text{one\_hot}(y_i)$, the

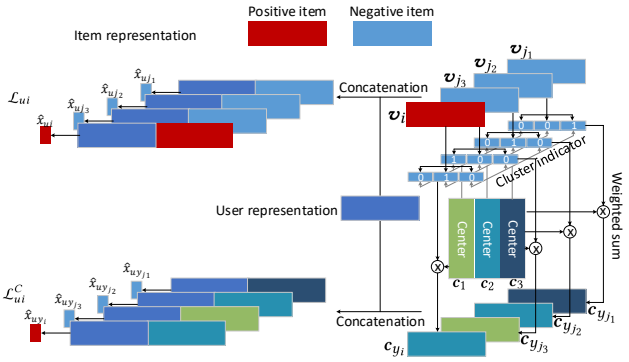


Figure 1: The joint learning framework for model with three negative samples and clustering with three centers.

argmax operator is then approximated by tempering Softmax:

$$b_i[k] \approx \hat{b}_i[k] = \frac{\exp(\text{sim}(\boldsymbol{v}_i, \boldsymbol{c}_k)/T)}{\sum_{k'} \exp(\text{sim}(\boldsymbol{v}_i, \boldsymbol{c}_{k'})/T)} \tag{17}$$

where T is a temperature parameter, controlling the degree of approximation. When $T \to 0$, the approximation becomes exact. Similar techniques have been applied in the Gumbe-Softmax trick [13, 20]. Given the approximation $\hat{\boldsymbol{b}}_i$ of one-hot representation of y_i, we have $\boldsymbol{c}_{y_i} = \boldsymbol{C}\boldsymbol{b}_i \approx \boldsymbol{C}\hat{\boldsymbol{b}}_i$, where $\boldsymbol{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_K]$ is a center matrix. Note that $\boldsymbol{C}\hat{\boldsymbol{b}}_i = \sum_k \boldsymbol{c}_k \hat{b}_i[k]$ can also be explained as a weighted sum of center vectors.

Due to importance of the temperature T for approximation, it should be carefully set or scheduled. If T is too large, $\hat{\boldsymbol{b}}_i$ is close to the uniform distribution, being far from the one-hot vector. If T is too small, not only the gradient vanishes but also $\hat{\boldsymbol{b}}_i$ is quite sensitive to the similarity. One practical strategy is to start with a high temperature, to ensure large gradient for parameter update at the beginning, and then cool it gradually with the training course, so that $\hat{\boldsymbol{b}}_i$ is closer and closer to the desired one-hot vector.

5 EXPERIMENT

We first compare the proposed algorithms with competing baselines, including the state-of-the-art algorithm. Following that, we study the effect of different sampling strategies on the recommendation accuracy. We finally perform sensitivity analysis with respect to the number of negative sampled items, embedding dimension, and the number of clusters.

5.1 Datasets

As showed in Table 1, five publicly available real-world datasets are used for evaluating the proposed algorithm. The datasets vary in difficulty of item recommendation, which may be indicated by the numbers of items and ratings, the density and concentration. The Amazon dataset is a subset of customers' ratings for Amazon books [21]. The rating scores are integers from 1 to 5, and books with scores higher than 4 are considered positive. The MovieLens dataset is from the classic MovieLens10M dataset. The rating scores are from 0.5 to 5 with 0.5 granularity, and movies with scores higher than 4 are considered positive. The CiteULike dataset collects users' personalized libraries of articles, the Gowalla dataset includes users'

Table 1: Dataset Statistics. Concentration indicates rating percentage on the top 5% most popular items.

	#users	#items	#rating	density	concentration
CiteULike	7,947	25,975	134,860	6.53e-04	31.56%
Gowalla	29,858	40,988	1,027,464	8.40e-04	29.15%
Amazon	130,380	128,939	2,415,650	1.44e-04	32.98%
Movielens	60,655	8,939	2,105,971	3.88e-03	61.98%
Echonest	217,966	60,654	4,422,471	3.35e-04	45.63%

check-ins at locations, and the Echonest dataset records users' play count of songs. In the Echonest dataset, we follow [11] to include songs to which a user listened at least 5 times as positive feedback. To ensure each user can be tested, we filter these datasets such that users rated at least 5 items. Note that the concentration in the Gowalla dataset is the smallest. This is because users often issue fake check-ins at nearby locations without any restriction.

For each user, we randomly sample his 80% ratings into a training set and the rest 20% into a testing test. 10% ratings of the training set are used for validation. We build recommendation models in the training set and evaluate them in the test set.

5.2 Metrics

The performance of recommendation is assessed by how well positive items in the test set are ranked. We use two widely-used metrics of ranking evaluation: Recall [12, 32] and NDCG [35]. Recall at a cutoff k, denoted as Recall@k, is the fraction of positive items in the top-k ranking list over the total number of positive items in the test set. NDCG at a cutoff k, denoted as NDCG@k, rewards method that ranks positive items in the first few positions of the top-k ranking list. The positive items ranked at low positions of ranking list contribute less than positive items at the top positions. The cutoff k in Recall and NDCG is set to 50 by default.

5.3 Parameter Settings

We implement the proposed algorithms PRIS in Keras with the Tensorflow backend, and optimize them with the Adam algorithm. Here we do not consider feature modeling, since it is not related to investigating effect of the new loss and samplers. The dimension of user embedding and item embedding is set to 32 by default. The batch size is fixed to 128 and the learning rate is fixed to 0.001 when evaluating PRIS in all datasets. The coefficient of activity regularization is tuned over $\{2, 1, 0.5\}$ in the validation set. Therefore, the proposed algorithm is easy-to-use and fast for hyperparameter tuning. It is also possible to set a much larger batch size for speeding up the training, but it may require to use a learning rate schedule, which starts with a larger value and decays gradually with a specific strategy like exponential or step decay until a much smaller preset value. The number of negative item to sample is set to 5 and the number of clusters in the cluster-based sampler is set to 100. Regarding the popularity-based sampler, we choose the popularity function as $f_i = \log(1 + c_i)$, where c_i denotes occurrence frequency of item i. This function has been observed to work better than another two options. The potential reason is that it is more likely for the adopted function to sample the long-tail items.

5.4 Baselines

We compare the proposed PRIS framework with the following baselines, whose embedding size is also set to 32.

- BPR [29], is a pioneer work of personalized ranking in recommender system. It uses pairwise logit loss, randomly samples a negative item and applies stochastic gradient descent (SGD) for optimization. It will be adapted to multiple negative items and to use mini-batch SGD for training. The number of negative items is set to 5, the regularization coefficient are tuned within $\{2, 1, 0.5\}$.
- AOBPR [28], improves BPR with adaptive oversampling. Following their settings, we set learning rate 0.05, epochs 256 and λ for the geometric distribution 500, and tune regularization coefficient within $\{0.005, 0.01, 0.02\}$. We use implementation in Librec[1] [7].
- WRMF [12, 27], weighted regularized matrix factorization, is designed for recommendation based on implicit feedback. It treats the data as indication of positive and negative preference associated with vastly varying confidence levels. This method has been considered as the state-of-the-art recommendation method for implicit feedback. The confidence level α for negative preference is tuned in the validation set within $\{1, 5, 10, 20, 50, 100, 200, 500\}$. The regularization coefficient is fixed to 0.01.
- WARP [36], uses the Weighted Approximate-Rank Pairwise loss function for collaborative filtering, which can perform better than WRMF in some datasets. The WARP loss is based on a ranking error function and extended to the online learning scenario. Given a positive item, it samples all items uniformly until a negative item is found. Based on the trials of sampling, the rank of the positive item can be estimated. We use the implementation in lightfm [15][2], where the maximal trial of sampling are set to 50 for efficiency. The regularization coefficient and the learning rate is tuned in the validation set within $\{0.05, 0.01, 0.005, 0.001\}$ and $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$, respectively.
- Self-Adversarial (SA) [30], is a recently proposed method for negative sampling based on the most recent recommendation model. This can be considered a special case of the proposed PRIS with the uniform sampling strategy, where back-propagation is not applied for sample weights.
- CML [11], Collaborative Metric Learning, learns a joint metric space to encode users' preference as well as user-user and item-item similarity. It also uses the WARP loss function for optimization. We use the authors' released code[3] and use the default parameters in the source code. The number of negative items in the sampler function is set to 20.

5.5 Comparison with Baselines

Table 2 shows the comparison results of PRIS (of the best sampler) with baselines. From them, we have the following key findings.

- *Finding 1 – The proposed algorithm consistently outperforms all baselines in all five datasets.* On average, the relative performance improvements are as least 19% with respect to NDCG@50 and 12% with respect to Recall@50. The relative improvements to WRMF are 23% on average with respect to NDCG@50 while the relative improvements to WARP can be up to 39%. In spite of the smallest

[1] https://www.librec.net/
[2] https://github.com/lyst/lightfm
[3] https://github.com/changun/CollMetric

performance lift compared to CML, the comparison with CML is not fair. This is because CML scans more items for negative sampling and it is based on Euclidean embedding instead of inner-product embedding. CML has been observed to outperform the inner-product based algorithm – WARP. This superiority can also be verified in the five datasets excluding Gowalla. Adapting the proposed algorithm to Euclidean embedding is a very promising future work. Moreover, it is also possible to improve sampling efficiency of CML and rank estimation of positive preferences based on the proposed samplers.

- *Finding 2 – Either of WRMF and WARP can not consistently beat the other; both of them show their own distinct strengths.* WARP shows better performance than WRMF in the datasets with lower concentration yet higher density. The best-performing dataset of WARP is Gowalla, in which the concentration is the lowest. This may be because it is much easier to sample negative items in the Gowalla dataset even when the model is trained a lot. Therefore, the concentration can be an indicator about which one of them is better. It is intuitive to deduce that the concentration is also correlated with the performance of popularity based recommendation. The results show that Recall@50 in the CiteULike, Gowalla, Amazon, MovieLens and Echonest dataset is 0.0609, 0.0493, 0.0240, 0.2921 and 0.0843 respectively. The recall in the Gowalla dataset is the second lowest, not the lowest because of the second highest density. This analysis may provide some clues for improving recommendation, such as ensemble of them.
- *Finding 3 – It is almost impossible for BPR to show comparable performance to WRMF.* This finding was also revealed in many existing literatures [17, 32, 33]. The main reason lies in the effectiveness of negative sampling. At the very beginning of training, the estimated scores of positive samples are close to negative samples, so the parameters can be significantly updated and the loss is dramatically reduced. After BPR is better trained, positive samples can be easily differentiated from the randomly sampled negative samples, so the gradient approaches zero and the parameters are almost not updated.
- *Finding 4 – Weighting negative samples with preference scores addresses small gradient magnitude of BPR to some extent.* This is based on the observation that SA outperforms BPR. SA and BPR use the same number of negative samples but distinguish from each other about whether weighting negative samples. In particular, SA places more emphasis on negative items with larger preference scores. Since the negative items with larger scores contribute more to gradient, imposing higher weight on them leads to much larger gradient than BPR.
- *Finding 5 – Effectiveness of negative sampler is important for personalized ranking.* SA uniformly samples items from unobserved ones, so the gradient vanish problem still exists. This problem can be addressed by PRIS since it is accompanied by a more effective negative sampler, which can sample higher-scored negative items with larger probability. Therefore, PRIS can keep ensuring a large gradient along the progress of training. This explains the reason why PRIS outperforms SA and motivates us to develop more advanced sampler in future work. This can also be deduced by the superiority of AOBPR to BPR in most cases. However, even with effective samplers, AOBPR can not show comparable performance to PRIS due to the absence of importance weighting.

5.6 Study of Sampling Strategies

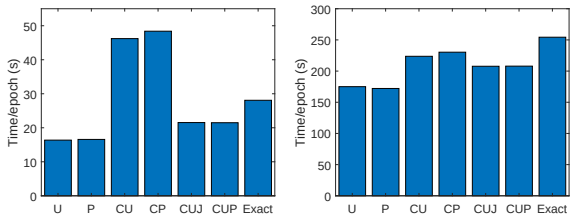
Following the comparison with baselines, we compare the effectiveness of different sampling strategies, whose results are reported in Table 3. From this table, we can reveal the following findings.

- *Finding 1 – The popularity-based sampler works surprisingly well.* The relative improvements to the uniform sampler are 8.67%, 4.03%, 11.48%, 5.41% and 7.51% with respect to NDCG@50 in the five datasets. This indicates that popular items with which a user does not interact are more likely to be negative. It is worth mentioning that the probability mass on the few most popular items should not be too large. Otherwise each time only the few most popular items are sampled, so the model may either under-fit or suffer from the false-negative problem. The improvement in the Gowalla dataset is the smallest. This may lie in the smallest concentration, so that popular items (except the few most popular ones) are least distinguishable from long-tail items compared to other datasets. It was reported in [28] the popularity-based sampler didn't lead to competitive quality. This finding can imply the advantage of the new ranking loss, which compensates deviation between the negative probability and the popularity distribution.
- *Finding 2 – Drawing negative samples based on the negative probability leads to large improvements.* This is reasonable, and consistent with our intuition, since these negative items can help to produce larger gradient for parameter update. However, computing preference scores for all items is very time-consuming particularly when there are a large number of items, so that it is seldom used in practice. This is also the reason that the results in two datasets with the most items are not reported. It is worth mentioning that "stochastic sampling" is a sufficient condition to ensure the improvement. If stochastic sampling is substituted with deterministic selection, the performance would dramatically deteriorate. This is because it severely suffers from the false-negative problem, where high-scored items are also likely to be positive (positive in the test set) rather than negative.
- *Finding 3 – Approximate sampling can be comparable and sometimes superior to exact sampling; approximate sampling alleviates the false-negative issue.* This is observed by comparing the exact sampler with cluster-based samplers. If items in each cluster are sampled based on the popularity rather than uniformly, the recommendation performance further improves. This is evidenced by comparing CP with CU in the table. One one hand, this implies that approximating item embedding with centers suffers from information loss. It is necessary to design advanced samplers in future work for reducing the information loss; on the other hand, item popularity can provide extra information to center approximation, compensating some of information loss.
- *Finding 4 – Learning-based clustering shows comparable recommendation performance to K-means but dramatically reduces clustering overhead.* Note that preference score of each item in cluster-based samplers is approximated by the center of item representation. Therefore, the samplers should be much more efficient than the exact sampler. However, the results reported in Figure 2 show that CU costs more time than Exact in the CiteULike dataset. This lies in clustering overhead in each iteration. The overhead can be reduced by incremental clustering, or lowering clustering frequency. Learning-based clustering can be considered incremental

Table 2: Comparison with baselines in the five datasets with respect to NDCG@50 and Recall@50.

	CiteULike		Gowalla		Amazon		MovieLens		Echonest		
	NDCG@50	IMP(%)	NDCG@50	IMP(%)	NDCG@50	IMP(%)	NDCG@50	IMP(%)	NDCG@50	IMP(%)	AVG_IMP
WRMF	0.1239	22.22	0.1329	29.34	0.0631	34.66	0.2958	7.71	0.1187	24.22	23.63
CML	0.1293	17.12	0.1245	38.06	0.0742	14.52	0.2606	22.26	0.1413	4.38	19.27
WARP	0.0878	72.58	0.1555	10.58	0.0560	51.75	0.2432	30.98	0.1111	32.76	39.73
BPR	0.1269	19.30	0.1216	41.37	0.0499	70.18	0.2460	29.49	0.0940	56.84	43.44
AOBPR	0.0990	52.90	0.1385	24.17	0.0563	50.80	0.2623	21.45	0.1040	41.86	38.23
SA	0.1328	14.07	0.1400	22.80	0.0629	35.03	0.2973	7.17	0.1196	23.28	20.47
PRIS	**0.1514**	-	**0.1719**	-	**0.0849**	-	**0.3186**	-	**0.1475**	-	-
	Recall@50	IMP(%)	Recall@50	IMP(%)	Recall@50	IMP(%)	Recall@50	IMP(%)	Recall@50	IMP(%)	AVG_IMP
WRMF	0.2710	15.46	0.2242	25.64	0.1445	24.53	0.5047	5.72	0.2219	16.92	17.65
CML	0.2775	12.77	0.2337	20.50	0.1593	12.94	0.4772	11.81	0.2530	2.58	12.12
WARP	0.1912	63.69	0.2650	6.29	0.1217	47.79	0.4440	20.17	0.2007	29.29	33.45
BPR	0.2772	12.89	0.2048	37.51	0.1171	53.65	0.4425	20.56	0.1829	41.90	33.30
AOBPR	0.2208	41.75	0.2369	18.90	0.1303	38.09	0.4780	11.63	0.2063	25.79	27.23
SA	0.2872	8.94	0.2355	19.58	0.1423	26.45	0.5145	3.70	0.2259	14.88	14.71
PRIS	**0.3129**	-	**0.2816**	-	**0.1799**	-	**0.5335**	-	**0.2595**	-	-

clustering, where centers are dynamically updated together with model parameters. Figure 2 shows the efficiency is indeed dramatically promoted. Interestingly, the recommendation performance is comparable or even sometimes better, showing the effectiveness of learning-based clustering.



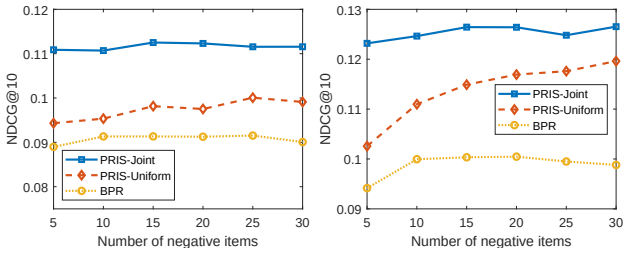
(a) CiteULike (b) Gowalla

Figure 2: The running time of PRIS with different samplers in the CiteULike and Gowalla dataset.

5.7 Sensitivity w.r.t The Number of Negative Samples

5.7.1 Settings. We vary the number of negative samples from 5 to 30 with a step 5, and report the results of PRIS-Joint with the Cluster-Popularity-Joint sampler, PRIS-Uniform with the Uniform sampler and BPR. The coefficient of activity regularization in these three models has been separately fine-tuned within $\{2, 1, 0.5\}$ while other hyperparameters are fixed by default.

5.7.2 Findings – PRIS is highly effective at exploiting a limited number of negative samples; when negative items are more informative, PRIS requires fewer samples. The results in the CiteULike and Gowalla dataset are shown in Figure 3. PRIS-Joint slightly improves when more negative items are provided. When supplying a uniform sampler, PRIS-Uniform improves more than PRIS-Joint with the growing number of negative items. On one hand, this shows that more complex sampler makes PRIS depend on fewer negative samples, indicating the superiority in terms of both optimization efficiency and recommendation effectiveness. On the other hand, without any advanced samplers, we can improve recommendation performance by feeding more negative samples. However, without weighting negative samples, BPR can not improve as much as PRIS-Uniform with the increasing number of negative samples. Therefore, even when many negative samples are considered, BPR can not put more emphasis on informative negative samples. This implies the proposed algorithm promotes the effectiveness of exploiting negative samples, by assigning larger weights to samples with higher preference scores.



(a) CiteULike (b) Gowalla

Figure 3: The effect of the number of negative items.

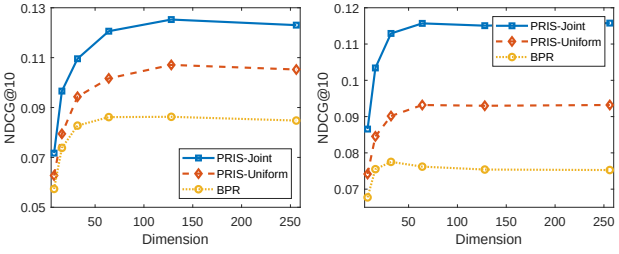
5.8 Sensitivity w.r.t Embedding Dimension

5.8.1 Settings. We vary the dimension of user representation and item representation in the set $\{8, 16, 32, 64, 128, 256\}$ and also report the results of PRIS-Joint, PRIS-Uniform and BPR. All the hyperparameters are fixed by default. Fine-tuning the coefficient of activity regularization may improve the performance, but the relative positions and the trends in the figure are not altered.

Table 3: The study of sampling strategies. "U", "P", "CU", "CP" denotes the "Uniform", "Popularity", "Cluster-Uniform", "Cluster-Popularity" sampler, respectively. "CUJ" and "CPJ" are samplers which are learned jointly with the model. The "Exact" sampler draws samples based on the negative probability $P(j|u,i)$ in Eq (2).

	NDCG@50					Recall@50				
Sampler	CiteULike	Gowalla	Amazon	MovieLens	Echonest	CiteULike	Gowalla	Amazon	MovieLens	Echonest
U	0.1328	0.1400	0.0629	0.2973	0.1196	0.2872	0.2355	0.1423	0.5145	0.2259
P	0.1443	0.1457	0.0701	0.3133	0.1286	0.3033	0.2420	0.1554	0.5315	0.2401
CU	0.1413	0.1660	0.0776	0.3102	0.1362	0.2993	0.2746	0.1676	0.5284	0.2460
CP	0.1505	0.1706	0.0824	0.3162	0.1440	0.3159	0.2803	0.1760	0.5320	0.2567
CUJ	0.1507	0.1720	0.0824	0.3177	0.1476	0.3135	0.2822	0.1762	0.5330	0.2611
CPJ	0.1514	0.1719	0.0849	0.3186	0.1475	0.3129	0.2816	0.1799	0.5335	0.2595
Exact	0.1428	0.1695	-	0.3143	-	0.3017	0.2811	-	0.5339	-

5.8.2 Findings – Models with the medium size are more powerful and better-performing, and are also better trained in the proposed framework. The results are reported in Figure 4. When embedding dimension increases, all of these three models begin a fairly steady climb toward the peak performance at around 128 dim, and follow by a slight drop. The performance gap between any two of them also shows a growing trend. This observation implies that the proposed framework also does better in training large models, particularly when the complex samplers are applied.



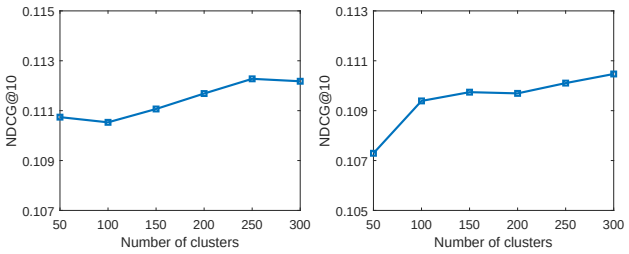
(a) CiteULike (b) Gowalla

Figure 4: The effect of embedding dimension.

5.9 Sensitivity w.r.t the Number of Clusters

5.9.1 Settings. The number of clusters are varied from 50 to 300 with a step 50. The hyperparameters are not fine-tuned since the overall trends are similar. The results of evaluation with the Cluster-Popularity sampler in the CiteULike and Gowalla dataset are showed in Figure 5.

5.9.2 Findings – More clusters can make cluster-based samplers better approximate the Exact sampler. Both figures show that the recommendation performance increases by clustering items into more groups. This may be because more clusters can provide more accurate approximation of item representations. For the sake of higher sampling efficiency, the number of clusters should not be too large. This results in comparatively small improvements of recommendation performance with the increase of cluster number. This motivates us to seek the alternative schemes of approximation, such as hierarchical clustering, to strike a better balance between representation capacity and sampling efficiency.



(a) CiteULike (b) Gowalla

Figure 5: The effect of the number of clusters.

6 CONCLUSION AND FUTURE WORKS

In this paper, we proposed a new ranking loss function based on importance sampling, which can put more emphasis on more informative negative samples. Therefore, the proposed algorithm addressed the problem of small gradient magnitude in BPR, in case of not resorting to any advanced samplers. We then designed four negative samplers from simple to complex based on rejection sampling. The static sampling using item popularity leads to more competitive quality than uniform sampling, indicating the advantage of the new ranking loss function compared to [28]. As informativeness of negative samples grows, the recommendation performance also improves and fewer negative samples are required. The extensive experiments with five real-world datasets of varied size and difficulty show that they can consistently beat the state-of-the-art item recommendation algorithms.

Several topics can be investigated in future work. First, though the cluster-based samplers perform better, the number of clusters should not be large due to efficiency. One way is to design a tree-based negative sampler. Alternative options include the methods in [4, 28] based on a mixture of multinoulli distributions. Second, it is also necessary to apply the proposed framework for metric learning and deep recommendation algorithms.

ACKNOWLEDGMENTS

The work was supported by grants from the National Natural Science Foundation of China (Grant No. 61976198, 61727809 and 61832017).

REFERENCES

[1] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of WWW'17*. 1341–1350.

[2] Guy Blanc and Steffen Rendle. 2018. Adaptive Sampled Softmax with Kernel Based Sampling. In *International Conference on Machine Learning*. 589–598.

[3] Jin Chen, Defu Lian, and Kai Zheng. 2019. Improving One-Class Collaborative Filtering via Ranking-Based Implicit Regularizer. In *Proceedings of AAAI'19*, Vol. 33. 37–44.

[4] Qin Ding, Hsiang-Fu Yu, and Cho-Jui Hsieh. 2019. A Fast Sampling Algorithm for Maximum Inner Product Search. In *The 22nd International Conference on Artificial Intelligence and Statistics*. 3004–3012.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[6] Ian J Goodfellow. 2014. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515* (2014).

[7] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. LibRec: A Java Library for Recommender Systems.. In *UMAP Workshops*, Vol. 4.

[8] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 297–304.

[9] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of SIGIR'16*, Vol. 16.

[10] Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14, 8 (2002), 1771–1800.

[11] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of WWW'17*. International World Wide Web Conferences Steering Committee, 193–201.

[12] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM'08*. IEEE, 263–272.

[13] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[14] Walid Krichene, Nicolas Mayoraz, Steffen Rendle, Li Zhang, Xinyang Yi, Lichan Hong, Ed Chi, and John Anderson. 2018. Efficient training on very large corpora via gramian estimation. *arXiv preprint arXiv:1807.07187* (2018).

[15] Maciej Kula. 2015. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439* (2015).

[16] Xiang Li, Tao Qin, Jian Yang, and Tie-Yan Liu. 2016. LightRNN: Memory and computation-efficient recurrent neural networks. In *Advances in Neural Information Processing Systems*. 4385–4393.

[17] Defu Lian, Yong Ge, Nicholas Jing Yuan, Xing Xie, and Hui Xiong. 2016. Sparse Bayesian Content-Aware Collaborative Filtering for Implicit Feedback. In *Proceedings of IJCAI'16*. AAAI.

[18] Defu Lian, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, Xing Xie, Tao Zhou, and Yong Rui. 2018. Scalable Content-Aware Collaborative Filtering for Location Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018). https://doi.org/10.1109/TKDE.2018.2789445

[19] Defu Lian, Kai Zheng, Yong Ge, Longbing Cao, Enhong Chen, and Xing Xie. 2018. GeoMF++ Scalable Location Recommendation via Joint Geographical Modeling and Matrix Factorization. *ACM Transactions on Information Systems (TOIS)* 36, 3 (2018), 1–29.

[20] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* (2016).

[21] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of KDD'15*. ACM, 785–794.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[23] Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*. 2265–2273.

[24] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model.. In *Aistats*, Vol. 5. Citeseer, 246–252.

[25] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. The MIT Press.

[26] Rong Pan and Martin Scholz. 2009. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 667–676.

[27] R. Pan, Y. Zhou, B. Cao, N.N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-class collaborative filtering. In *Proceedings of ICDM'08*. IEEE, 502–511.

[28] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of WSDM'14*. ACM, 273–282.

[29] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI'09*. AUAI Press, 452–461.

[30] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[31] Gábor Takács and Domonkos Tikk. 2012. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 83–90.

[32] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of KDD'11*. ACM, 448–456.

[33] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of KDD'15*. ACM, 1235–1244.

[34] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 515–524.

[35] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. 2007. Maximum margin matrix factorization for collaborative ranking. *Proceedings of NIPS'07* (2007), 1–8.

[36] Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning* 81, 1 (2010), 21–35.

[37] Hsiang-Fu Yu, Mikhail Bilenko, and Chih-Jen Lin. 2017. Selection of negative samples for one-class matrix factorization. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 363–371.

[38] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 785–788.