CoSam: An Efficient Collaborative Adaptive Sampler for Recommendation

JIAWEI CHEN, Zhejiang University
CHENGQUAN JIANG, CAN WANG, SHENG ZHOU, YAN FENG, and CHUN CHEN, Zhejiang University
MARTIN ESTER, Simon Fraser University
XIANGNAN HE, University of Science and Technology of China

Sampling strategies have been widely applied in many recommendation systems to accelerate model learning from implicit feedback data. A typical strategy is to draw negative instances with uniform distribution, which, however, will severely affect a model's convergence, stability, and even recommendation accuracy. A promising solution for this problem is to over-sample the "difficult" (a.k.a. informative) instances that contribute more on training. But this will increase the risk of biasing the model and leading to non-optimal results. Moreover, existing samplers are either heuristic, which require domain knowledge and often fail to capture real "difficult" instances, or rely on a sampler model that suffers from low efficiency.

To deal with these problems, we propose CoSam, an efficient and effective collaborative sampling method that consists of (1) a collaborative sampler model that explicitly leverages user-item interaction information in sampling probability and exhibits good properties of normalization, adaption, interaction information awareness, and sampling efficiency, and (2) an integrated sampler-recommender framework, leveraging the sampler model in prediction to offset the bias caused by uneven sampling. Correspondingly, we derive a fast reinforced training algorithm of our framework to boost the sampler performance and sampler-recommender collaboration. Extensive experiments on four real-world datasets demonstrate the superiority of the proposed collaborative sampler model and integrated sampler-recommender framework.

CCS Concepts: • **Information systems** → **Recommender systems**;

Additional Key Words and Phrases: Sampling, recommendation, efficiency, adaption

ACM Reference format:
Jiawei Chen, Chengquan Jiang, Can Wang, Sheng Zhou, Yan Feng, Chun Chen, Martin Ester, and Xiangnan He. 2021. CoSam: An Efficient Collaborative Adaptive Sampler for Recommendation. *ACM Trans. Inf. Syst.* 39, 3, Article 34 (May 2021), 24 pages.
https://doi.org/10.1145/3450289

This work was supported by the National Key R&D Program of China (grants 2018AAA0101505 and 2019YFB1600700) and the National Natural Science Foundation of China (grant U1866602).
Authors' addresses: J. Chen, Zhejiang University & University of Science and Technology of China, China; email: sleepyhunt@zju.edu.cn; C. Jiang, Can Wang (corresponding author), S. Zhou, Y. Feng, and C. Chen, Zhejiang University, 38 ZheDa Road, Hangzhou, China; emails: imjcqt@gmail.com, {wcan, zhousheng_zju, fengyan}@zju.edu.cn, chenc@cs.zju.edu.cn; M. Ester, Simon Fraser University, 8888 University Drive, Burnaby, Canada; email: ester@cs.sfu.ca; X. He, University of Science and Technology of China, 96 JinZhai Road, Hefei, China; email: xiangnanhe@gmail.com.

1046-8188/2021/05-ART34 $15.00
https://doi.org/10.1145/3450289

1 INTRODUCTION

Recent research attention on recommendation is increasingly shifting from explicit feedback data (a.k.a. rating data) toward implicit feedback data. Implicit feedback data is often a natural byproduct of users' behavior (e.g., clicks, purchase, check-in), and thus is more prevalent, inexpensive and scalable. However, recommendation using implicit feedback is more challenging, as it only provides partial feedback signal (a.k.a. one-class feedback)—only positive feedback are observed. Although the unobserved feedback data can be treated as a source of negative signal, its large scale incurs expensive time cost in learning a recommendation model. Existing methods mainly rely on the stochastic gradient descent optimizer and negative sampling strategies to speed up training procedures. Since the gradients are estimated on the sampled instances, sampling strategies play an important role in recommendation systems and significantly affect the training efficiency and recommendation accuracy.

There is rich literature on sampling strategies for better recommendation; however, to our knowledge, existing samplers lack one or more desiderata. They can be roughly classified into the following four types:

- *Uniform sampler*, the most popular strategy that samples negative instances with equal probability. Although efficient, it usually causes slow convergence and high variance, leading to non-optimal results [35, 50, 51].
- *Heuristic-based samplers*, which define specific sampling rules to over-sample the "difficult" (a.k.a. informative) negative instances. The intuition behind this type of sampler is that the "difficult" instances, which can be defined as the items with higher popularity [50, 52] or higher-ranking position [35, 52], make more contribution on gradients. However, these methods involve heuristic alterations to the model and data, requiring recommendation expertise or tedious hyper-parameter tuning. Moreover, heuristic samplers usually lack flexibility and often fail to capture really "difficult" instances.
- *Model-based adaptive samplers*, which define a class of sampling distribution with a flexible sampler model [10, 13, 34, 42, 43]. Existing model-based samplers mainly adopt an adversarial learning framework [15], where the sampler and the recommender play a min-max game. With adversarial learning between the two models, the sampler will learn to fit users' preference distribution and thus adaptively generate positive-like "difficult" instances. Although this type of sampler has been proved effective, it tends to be very slow in a large dataset. The reason is that the sampling distribution will dynamically evolve with the training proceeding, and the sampling process usually requires a traversal of all instances, which is time consuming and computationally inefficient. Moreover, the adversarial sampler will yield a biased estimate of the gradients since its uneven sampling distribution will skew the instance contribution, resulting in sub-optimal results. Typically, the recommendation model will be biased to over-fit the "difficult" negative instances. These instances will be predicted with relatively low preference scores. In fact, these "difficult" but positive-like items are likely to be adopted by the user.
- *Auxiliary information-based samplers*, which leverage auxiliary information (e.g., social network, users' exposure) to sample informative negative instances [5, 10]. However, these auxiliary information may not be available in many recommender systems.

Designing a satisfactory sampler is important but challenging. Drawing on the advantages of existing model-based samplers, we explore a more advanced model-based sampler. We have to address the following key research problems:

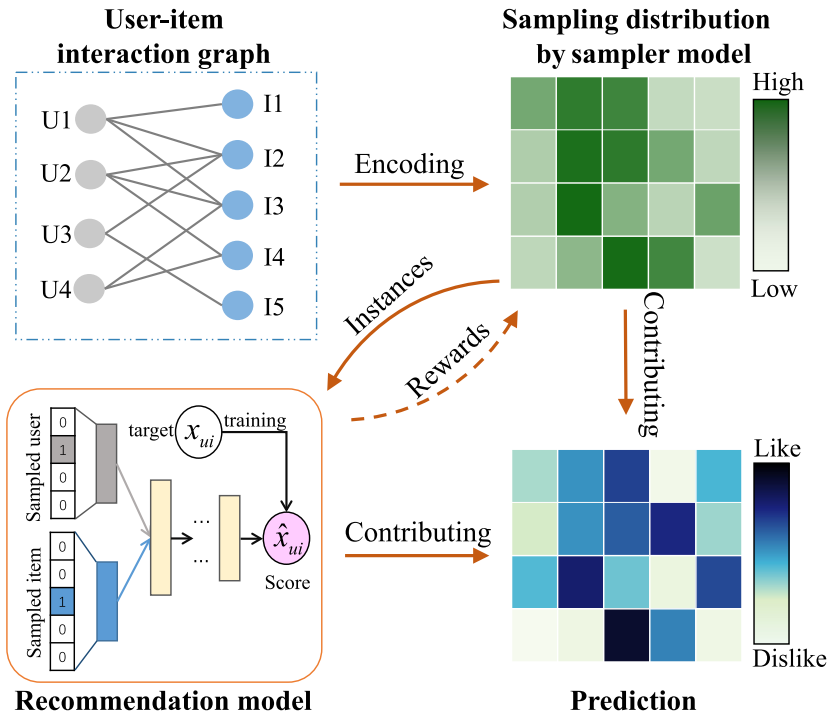


Fig. 1. Illustration of CoSam, which leverages an interaction graph into sampling and integrates sampling distribution (model) into prediction. The user-item interaction graph is constructed from user-item interaction information, where we give links for the user-item pair with positive feedback.

(P1) How do we design a sampler model that supports efficient and effective sampling?
(P2) How do we mitigate the bias caused by the uneven sampling probability?

In this article, we propose CoSam (*Co*llaborative *Sam*pler), a novel model-based adaptive sampler. Corresponding to the preceding two problems, CoSam has two improvements and "collaborative" has twofold meanings. First, to tackle problem (P1), we propose a novel *collaborative* sampler model, which leverages *collaborative* signals in sampling. The collaborative signals—more specifically, the user-item interaction information and its extended user-item interaction graph as illustrated in Figure 1—reveals correlations among users and items. These correlations can be expressed with the information paths along the graph. For example, as illustrated in Figure 1, the information paths U1-I2-U2, U1-I3-U2 between users U1 and U2, indicate the users have similar behaviors, as they have interacted with common items I2, I3. Correspondingly, their feedback data may have similar level of difficulty and should be sampled with similar distribution. Another example can be seen from the rich information paths between user U1 and item I4 (e.g., U1-I2-U2-I4, U1-I3-U4-I4). It reflects that U1 and I4 may have high affinity although they have no directed interaction, since U1's similar users U2,U4 have interacted with I4. Correspondingly, this feedback data is potentially highly informative (difficult) for the recommendation model. Thus, CoSam defines a sampler model on the graph and constructs adaptive sampling distribution based on information paths, which naturally encodes this useful correlation information into distribution and potentially boosts sampling performance. Moreover, a specific random walk–based sampling strategy is developed to support fast sampling from the proposed dynamic distribution, which will evolve over the large item space as training goes on. Our theoretical analysis proves that the CoSam

satisfies the desirable properties of normalization, adaption, interaction information awareness, and sampling efficiency.

Second, to address problem (P2), we propose a novel integrated sampler-recommender framework to replace the adversarial framework. In our integrated framework, the sampler acts as a filter to remove relatively "easy" instances and the recommender focuses on differentiating the really "difficult" instances that cannot be differentiated by the sampler. This way, the sampler and the recommender will *collaboratively* characterize users' preference and satisfy the following desirable property: even although the recommender model is learned with a highly uneven sampler, its biased prediction can be refined by integrating the sampling probability. We further derive a fast reinforced training algorithm of our framework to boost the sampler's performance and sampler-recommender *collaboration*. Our framework and algorithm can be used for most recommender models.

It is worthwhile to highlight the following contributions:

- We propose an efficient and effective collaborative sampler model by leveraging user-item interaction information into sampling.
- We propose an integrated sampler-recommender framework, which leverages the sampler model into prediction to offset the sampling bias.
- Our experimental evaluations on four well-known benchmark datasets validate the superiority of the proposed collaborative sampler model and integrated sampler-recommender framework.

The rest of this article is organized as follows. We give the problem definition and background in Section 2. The proposed sampling method, CoSam, is introduced in Section 3. The experimental results and discussions are presented in Section 4. We briefly review related works in Section 5. Finally, we conclude the article and present some directions for future work in Section 6.

2 PRELIMINARIES

In this section, we first give the problem definition of the implicit recommendation. Then, we introduce the model-based sampler with the adversarial sampler-recommender framework and analyze its merits and weaknesses.

2.1 Recommendation Using Implicit Feedback

Suppose we have a recommender system with user set U (including n users) and item set I (including m items). The implicit feedback data is represented as an $n \times m$ binary matrix $\mathbf{X}$ with entries $x_{ui} \in \{0, 1\}$ denoting whether or not user u has interacted by item i. In addition, we let $\mathcal{X}_u$ denote the set of positive items that have been interacted by user u. The task of an implicit recommender system can be stated as follows: learning a user's preference from the available implicit feedback and recommending items that are most likely to be interacted by the user. The notations of this work are summarized in Table 1.

2.2 Adversarial Sampler-Recommender Framework

The adversarial sampler-recommender framework contains a sampler model (S), a recommender model (R), and weaknesses, which are discussed next.

Sampler model (S). $G_u \sim Multinomial(N_u, p_s(i|u; \theta))$, which draws a small item set (G_u) for each user u from the global item space, where the parameter N_u denotes the number of sampled items and the conditional probability $p_s(i|u; \theta)$ denotes the flexible sampling distribution with parameters θ. The sampler model tries to fit the user's preference distribution to generate "difficult"

Table 1. Notation and Definitions

Notation	Annotation
U	User set
I	Global item set
$\mathbf{X}$	The matrix of implicit feedback data
$\mathcal{X}_u$	The positive item set of user u
S	The sampler model
R	The recommender model
G_u	The sampled item set for each user u returned by S
$N_u, p_s(i\|u;\theta)$	The sample size and sampling distribution of S for each user u
$f_r(u,i;\varphi)$	A parameterized function mapping user/item attributes into predicted preference scores
$\gamma_i^{(t)}$	The item label vector for item i after t-th revisions in CoSam
$\rho_u^{(t)}$	The sampling distribution for each user u after t-th revisions in CoSam
z_i	The one-hot vector in which the i-th element is equal to 1
$\mathcal{U}$	The uniform distribution in which each element is set equally

instances for the recommender. We remark that some works [10] will remove the sampled positive instances from G_u.

Recommender model (R). This model captures the user's preference and makes recommendation. In the adversarial framework, R is optimized to distill real positive items ($i \in \mathcal{X}_u$) from the sampled training set $[\mathcal{X}_u, G_u]$: $\mathcal{X}_u \sim p_r(\mathcal{X}_u|[\mathcal{X}_u, G_u], f_r(u,i;\varphi))$, where $[\mathcal{X}_u, G_u]$ denotes the union of the positive item set X_u and sampled negative item set G_u. $f_r(u,i;\varphi)$ is a parameterized preference function, mapping attributes (e.g., ids) of users and items into the predicted preference with parameters φ; p_r denotes likelihood function. Without loss of generality, here we choose the Bernoulli classifier as an example for further derivation, and it can be easily replaced by other functions (e.g., Gaussian likelihood [33], Bayesian Personalized Ranking [36]). Formally, we have

$$p_r(\mathcal{X}_u|[\mathcal{X}_u, G_u]) = \prod_{i\in[\mathcal{X}_u, G_u]} Bernoulli(x_{ui}, f_r(u,i;\varphi)). \tag{1}$$

In the adversarial framework, S and R play an adversarial game and optimize the following objective function:

$$\min_{\theta}\max_{\varphi} \sum_{u\in U} E_{G_u\sim S}\left[\sum_{i\in X_u} \log(f_r(u,i;\varphi)) + \sum_{i\in G_u} \log(1-f_r(u,i;\varphi))\right], \tag{2}$$

where $E_{G_u\sim S}[.]$ stands for $E_{G_u\sim Multinomial(N_u, p_s(i|u;\theta))}[.]$. Naturally, with the competition between two models, S will be learned to draw the difficult instances with higher preference scores, which provide larger gradients and more information to train R.

Weaknesses. However, although the model-based sampler with adversarial sampler-recommender framework is capable of adaptively capturing difficult instances, we emphasize two weaknesses of this type of method. First, the sampling process is time-cost expensive. Since the sampling probability will evolve over the large item space with the training going on, the sampling process requires a traversal of all instances (($O(n \times m)$)) to update the dynamic sampling distribution and to draw instances from the distribution. Second, the adversarial uneven sampler will create biases on the recommendation model, leading to non-optimal results [4]. It can be understood from the expected objective function of the recommender model in the adversarial

framework:

$$\sum_{u \in U} \left[\sum_{i \in \mathcal{X}_u} \log(f_r(u, i; \varphi)) + \sum_{i \in I} N_u E[p_r(i|u)] \log(1 - f_r(u, i; \varphi)) \right], \tag{3}$$

which is different from the original objective function without sampling. The negative instances have been weighted with sampling probability, and the objective function is biased from the data likelihood. Moreover, with the adversarial training between the sampler and recommender, the sampler will highly incline to draw "positive-like" negative data [14, 15]. Correspondingly, these instances will be over-trained and the learned preference scores $f_r(u, i; \varphi)$ of these instances will be biased to have relatively low values. In fact, these negative but "positive-like" items are highly likely to be interacted by the user. The recommendation performance will suffer. Meanwhile, the prediction accuracy on other relatively "easy" instances may deteriorate too due to the insufficient training of them. A naive solution for this problem is to offset the training instances with the inverse of their sampling probability. However, the gradients will be highly instable and potentially exploded.

Thus, in this article, we are interested in and propose a solution to two related problems:

(1) A new sampler model that supports efficient and effective sampling.
(2) A new sampler-recommender framework to mitigate biases caused by an uneven sampler.

3 METHODOLOGY

We now present the proposed sampler CoSam. There are two key components in CoSam: a collaborative sampler model and an integrated sampler-recommender framework.

3.1 Collaborative Sampler Model

To support both effective and efficient sampling, we propose a new sampler model based on a collaborative signal, which contains rich correlation information among users and items. We would like to encode such useful information into a sampling distribution and thus construct an interaction graph (Figure 2) where we give links for the user-item pairs with positive feedback. Inspired by the success of the graph convolutional network model [12, 28, 44, 49, 57] in capturing graph proximity, CoSam contains many propagation layers, which refine users' sampling probability based on their graph neighbors. Let vector ρ_u denote the sampling distribution for each user u over items ($p_s(i|u; \varphi) \equiv \rho_u$). We initially set $\rho_u^{(0)}$ with uniform distribution $\mathcal{U}$ and set the initial label for each item with its item indicator vector $\gamma_i^{(0)} = \mathbf{z}_i$. $\mathbf{z}_i$ is an one-hot vector in which the i-th element is equal to 1. The model will propagate user and item information along the graph to refine users' sampling distribution and naturally inject useful correlation information into the sampling distribution. Concretely, as shown in Figure 2, in each layer, each node in the graph collects information from the connected neighbors and reconstructs their distribution or labels as follows:

$$\rho_u^{(t)} = c_1 \sum_{i \in I, i \in \mathcal{N}_u} w_{ui}^{(1)} \gamma_i^{(t-1)} + (1 - c_1) \rho_u^{(0)}, \tag{4}$$

$$\gamma_i^{(t)} = c_2 \sum_{u \in U, u \in \mathcal{N}_i} w_{iu}^{(2)} \rho_u^{(t-1)} + (1 - c_2) \gamma_i^{(0)}, \tag{5}$$

where $\mathcal{N}_u$ ($\mathcal{N}_i$) denotes the connected item (user) set of user u (item i). The parameters c_1, c_2 ($0 \le c_1, c_2 \le 1$) specify the relative contributions from the neighbors and the initial vectors.

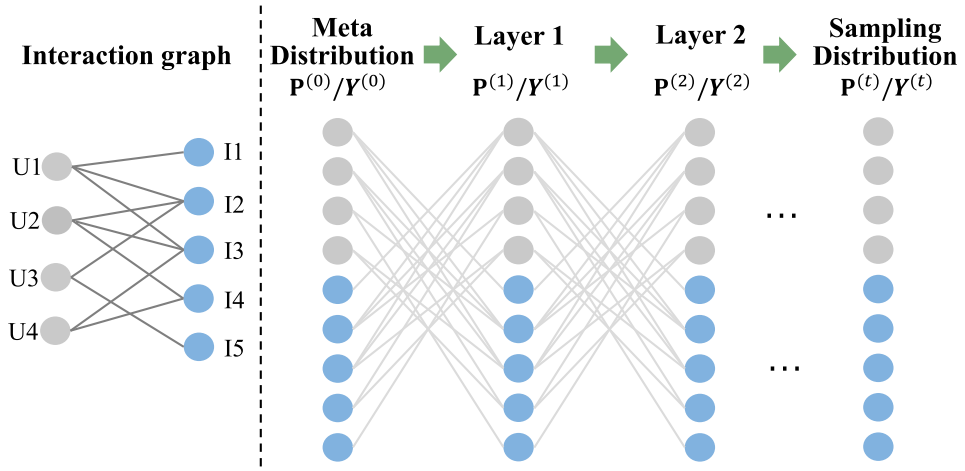


Fig. 2. Illustration of the collaborative sampler model.

$w_{ui}^{(1)}, w_{iu}^{(2)}$ are learned parameters, which balances the heterogenous influence from different neighborhoods and meets $\sum_{i\in I, i\in\mathcal{N}_u} w_{ui}^{(1)} = 1$, $\sum_{u\in U, u\in\mathcal{N}_i} w_{iu}^{(2)} = 1, w_{ui}^{(1)} > 0, w_{ui}^{(2)} > 0$. Overall, CoSam models sampling probability with an interactive graph-aware function g parameterized by edge weights w ($w^{(1)}, w^{(2)}$), to which Equations (4) and (5) converge:

$$\begin{bmatrix}\mathbf{P}\\ \mathbf{Y}\end{bmatrix} = g_w(\mathbf{X}) \equiv \lim_{t\to\infty}\begin{bmatrix}\mathbf{P}^{(t)}\\ \mathbf{Y}^{(t)}\end{bmatrix} = \left(\mathbf{I} - \mathbf{C}\begin{bmatrix}\mathbf{0} & \mathbf{W}^{(1)}\\ \mathbf{W}^{(2)} & \mathbf{0}\end{bmatrix}\right)^{-1}(\mathbf{I}-\mathbf{C})\begin{bmatrix}\mathbf{P}^{(0)}\\ \mathbf{Y}^{(0)}\end{bmatrix}, \tag{6}$$

where we collect sampling distribution ρ_u for each user as matrix $\mathbf{P}$ and collect labels γ_i for each item as matrix $\mathbf{Y}$; collect edge weights $w_{ui}^{(1)}, w_{iu}^{(2)}$ as matrixes $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$, in which $\mathbf{W}_{ui}^{(1)} = w_{ui}^{(1)}$, $\mathbf{W}_{iu}^{(2)} = w_{iu}^{(2)}$ for connected user-item pairs and $\mathbf{W}_{ui}^{(1)} = 0, \mathbf{W}_{iu}^{(2)} = 0$ for others; and collect c_1, c_2 as a $(n+m)\times(n+m)$ diagonal matrix $\mathbf{C}$, in which the first n diagonal elements are c_1 and other diagonal elements are c_2. We show that the proposed collaborative sampler model satisfies the following four desirable properties: normalized, interaction information awareness, sampling efficiency, and adaption.

Normalized. The constructed sampling model must have a valid probability distribution over the item space—that is, for each user u, we have $\sum_{i\in I}\rho_{ui} = 1$.

Proof. Let us mark the adjacency matrix of the interactive graph as $\mathbf{W} = \left[\begin{smallmatrix}\mathbf{0} & \mathbf{W}^{(1)}\\ \mathbf{W}^{(2)} & \mathbf{0}\end{smallmatrix}\right]$. Here we mark user nodes in the interaction graph with ids from 1 to n and item nodes with ids from $n+1$ to $n+m$. Let $\mathbf{e}$ denote the column vector of $n+m$ ones. We have $(I-\mathbf{CW})\mathbf{e} = (I-\mathbf{C})\mathbf{e}$. Note that $(I-\mathbf{CW})^{-1}(I-\mathbf{CW})\mathbf{e} = \mathbf{e}$. Thus, we can get $(I-c\mathbf{W})^{-1}(I-\mathbf{C})\mathbf{e} = \mathbf{e}$. Then, we have $\left[\begin{smallmatrix}\mathbf{P}^{(t)}\\ \mathbf{Y}^{(t)}\end{smallmatrix}\right]\mathbf{e} = (I-c\mathbf{W})^{-1}(I-\mathbf{C})\left[\begin{smallmatrix}\mathbf{P}^{(0)}\\ \mathbf{Y}^{(0)}\end{smallmatrix}\right]\mathbf{e} = \mathbf{e}$. □

Interaction information awareness. As we can see from Equation (6), the matrix $\mathbf{H} = (I-CW)^{-1}(I-C)$ is a graph or diffusion kernel [56], which has been widely adopted to measure nodes proximity in the graph. Thus, as shown in Figure 3, the constructed sampling distribution can be considered as a mixture of the meta distributions (uniform distribution or item indictor distribution), where the mixing coefficients are specified with the adaptive interaction

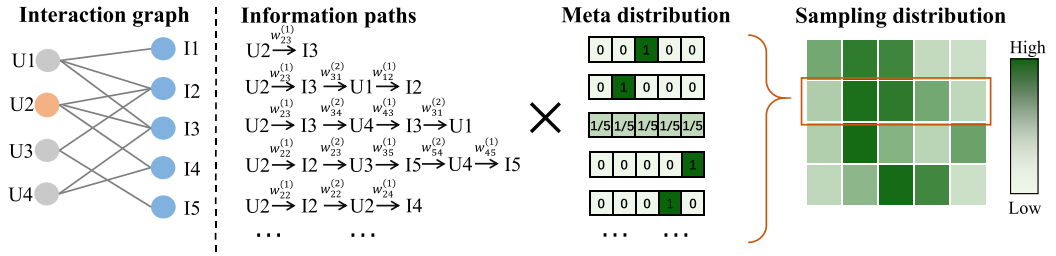


Fig. 3. The constructed sampling distribution of CoSam can be considered as a mixture of meta distributions, where the mixing coefficients are controlled by the information paths and the learned path strength.

graph proximity, which will evolve with the training process going on. In other words, we have the following formula of the sampling distribution of user u:

$$\rho_u = \sum_{k \in U} \mathbf{H}_{u,k} \mathcal{U} + \sum_{k \in I} \mathbf{H}_{u,n+k} \mathbf{z}_k, \tag{7}$$

where $\mathbf{H}_{i,j}$ denotes the (i,j)-th element in the matrix $\mathbf{H}$ characterizing the graph proximity between nodes i and j. The graph proximity $\mathbf{H}_{i,j}$ can be further depicted as aggregated strength of different information paths between nodes i and j:

$$\begin{aligned} \mathbf{H}_{ij} &= \Big((1-c_1) + (1-c_2)c_1 \mathbf{W}_{i,j} + (1-c_1)c_1 c_2 \prod_{k_1 \in I} \mathbf{W}_{i,k_1} \mathbf{W}_{k_1,j} \\ &\quad + (1-c_2)c_1 c_2 c_1 \prod_{k_1 \in I, k_2 \in U} \mathbf{W}_{i,k_1+m} \mathbf{W}_{k_1+m,k_2} \mathbf{W}_{k_2,j} + \dots \Big) \\ &= \sum_{\mathcal{P} \in \mathcal{A}_{ij}} F(\mathcal{P}), \end{aligned} \tag{8}$$

where $\mathcal{P}$ denotes a path instance on the graph and $\mathcal{A}_{ij}$ denotes the set of path $\mathcal{P}$ between nodes i and j. $F(\mathcal{P})$ is the product of edge weights on the path, which can be defined as the path strength between the nodes. The path with shorter length and larger edge weights will have stronger path strength. Thus, as illustrated in Figure 3, the constructed sampling probability can be considered as a mixture of meta distributions, where the mixing coefficients are controlled by the number of information paths and adaptive paths strength. A closer user-item pair with more and stronger information paths will be sampled with higher probability. Naturally, the rich interaction information has been encoded into the sampling distribution.

Sampling efficiency. Although the sampling distribution of CoSam will evolve with training going on, CoSam supports fast sampling with the following **adaptive random walk strategy (ARW)**:

Adaptive random walk strategy: For the target user u, we perform the adaptive random walk from u to sample the instances from ρ_u: at each step of random walk, we are at a certain node v. We have two options. First, if v is a user (or item) node, with probability $(1-c_1)$ (or $(1-c_2)$), we terminate the random walk. We randomly sample an item with meta distribution (or select v as an instance). Second, with probability c_1 (or c_2), we randomly transfer to one of v's neighbors based on current learned weights W_{v*}.

It is easy to check that the path strength is equal to the sampled probability of the walking path from the aforementioned ARW: $F(\mathcal{P}|\mathcal{P} \in \mathcal{A}_{uv}) = p_{ARW}(\mathcal{P}, v|u, \theta)$. Thus, the sampled items with ARW meet distribution ρ_u, even though ρ_u will evolve with training going on. This way, the

sampling process just requires a random walk along the graph instead of an expensive traversal of all items. The complexity of sampling an instance can be heavily reduced from $O(m)$ to $O(l_r)$, where l_r denotes the the length of the random walk and its expectation is upper bounded with $E[l_r] = 2/(1 - min(c_1, c_2))$. In practical terms, we are usually interested in users' local graph contexts and thus control the length of random walk smaller than 10, which is much smaller than the number of items.

We remark that the training and updating of the collaborative sampler is efficient too, which will be discussed in Section 3.3.

Adaption. We describe the adaption of our collaborative sampler model in the following two aspects. First, the sampling distribution of CoSam is defined with a parameterized function instead of pre-defined values. Thus, the sampling distribution will adaptively evolve with the training process going on. Second, the random walk–based sampling strategy has adaptive transfer probability so that it supports adaptively sampling from dynamic sampling distribution.

3.2 Integrated Sampler-Recommender Framework

Here we present our novel integrated sampler-recommender framework. Instead of letting the sampler (S) and recommender (R) play an adversarial game, we integrate them into a probabilistic generative process. The framework is motivated by the recent business study of the consumer behavior. As suggested in the work of Bray [3], consumers will first collect a set of potentially preferred candidate items and then carefully select desired items from the set. Correspondingly, S and R are integrated into the following decision-making process:

(1) Sample a small potential-preferred item set G_u from the global item set based on S:
$G_u \sim Multinomial(N_u, p_s(i|u;\theta))$.
(2) Distill truly positive items from G_u based on R:
$\mathcal{X}_u \sim p_r(\mathcal{X}_u|G_u, f_r(u,i;\varphi))$.

The intuition behind this process is similar to the adversarial framework: S acts as a filter to remove relatively "easy" instances, and thus R can be guided to focus on the really "difficult" instances that cannot be differentiated by S. But the integrated framework differs from the adversarial framework in that the sampler in our framework will make a contribution on prediction. S and R in our framework collaboratively characterize users' preference. Concretely, the probability of the item being positive can be depicted as follows:

$$\begin{aligned} p(i \in \mathcal{X}_u) &= p(i \in G_u)p(i \in \mathcal{X}_u | i \in G_u) \\ &\propto p_s(i|u;\theta) f_r(u,i;\varphi). \end{aligned} \tag{9}$$

A good property of our framework can be observed: although the recommender model is trained with a highly uneven sampler, its biased prediction can be refined by integrating the sampling probability. It is coincident with our intuition. R's biased prediction on the "difficult" instances, which may be biased with low values due to the over-training, will be refined by multiplying by a relatively large value; however, the prediction on these "easy" instances, which is highly negative, will be weighted with small values. Equation (9) guarantees that the final prediction will fit the data likelihood.

3.2.1 Training Objective and Algorithm. We further derive a fast and general training algorithm to learn the sampler model (S) and recommender model (R) in our framework. To make unbiased

prediction, S and R are optimized by maximizing the following data likelihood:

$$\begin{aligned}\log p(X) &= \sum_{u \in U} \log p(\mathcal{X}_u) = \sum_{u \in U} \log \sum_{G_u} p(\mathcal{X}_u, G_u) \\ &= \sum_{u \in U} \log \sum_{G_u} p(\mathcal{X}_u | G_u) p(G_u).\end{aligned} \tag{10}$$

However, directly optimizing marginal probability $p(X)$ is intractable due to the sum over all possible sampled sets inside the logarithm. The number of the feasible item sets is exponential growth, which causes an efficiency problem. To deal with this problem, we do some derivations of Equation (10) for fast learning.

Note that the positive instances must be contained in the sampled set, as otherwise they cannot be selected by model R. Thus, we can divide the G_u into two disjoint parts: the positive item set G_u^+ , which is observed constant and equivalent to $\mathcal{X}_u$, and the rest of item set G_u^r containing other sampled items. Thus, the probability of the sampled item set can be approximated as $p(G_u) \approx C_{N_u}^{|\mathcal{X}_u|} p(G_u^+) p(G_u^r)$, where $C_{N_u}^{|\mathcal{X}_u|}$ denotes combinatorial numbers. We have the following:

$$\begin{aligned}\log p(X) &= \sum_{u \in U} \log \sum_{G_u} p(\mathcal{X}_u, G_u) \\ &= \sum_{u \in U} \log \sum_{G_u^r} p\left(G_u^+\right) p\left(G_u^r\right) p\left(\mathcal{X}_u | G_u^+, G_u^r\right) + const \\ &= \sum_{u \in U} \left(\log p\left(G_u^+\right) + \log E_{G_u}\left[p\left(\mathcal{X}_u | G_u^+, G_u^r\right)\right]\right) + const,\end{aligned} \tag{11}$$

where *const* denotes the constant term and can be left out. However, directly optimizing Equation (11) is intractable too due to the expectation over the G_u^r inside the logarithm. Thus, we turn to optimize the following tight lower bound based on Jensen's inequality:

$$\begin{aligned}\log p(X) &\geq \sum_{u \in U} \left(\log p(G_u^+) + E_{G_u}\left[\log p\left(\mathcal{X}_u | G_u^+, G_u^r\right)\right]\right) \\ &= \sum_{u \in U} \left(\sum_{i \in \mathcal{X}_u} \log p_s(i|u;\theta) + \sum_{i \in \mathcal{X}_u} \log f_r(u,i;\varphi) + E_{G_u}\left[\sum_{i \in G_u} (1 - x_{ui}) \log(1 - f_r(u,i;\varphi))\right]\right) \\ &= \sum_{u \in U} \left(\sum_{i \in \mathcal{X}_u} \log p_s(i|u;\theta) + \sum_{i \in \mathcal{X}_u} \log f_r(u,i;\varphi) + N_u E_{p_s}[(1 - x_{ui}) \log(1 - f_r(u,i;\varphi))]\right).\end{aligned} \tag{12}$$

Optimizing the recommendation model. By dropping irrelevant terms in Equation (12), we have the following objective function of R:

$$L_\varphi = \sum_{i \in \mathcal{X}_u} \log f_r(u,i;\varphi) + \sum_{\alpha \in G_u, G_u \sim S} (1 - x_{u\alpha}) \log(1 - f_r(u,\alpha;\varphi)), \tag{13}$$

where we perform a sampling approximation in which item α is from the sampled item set G_u returned by S. We remark that the training objective of R in our framework is consistent with an adversarial framework. R is trained on the observed positive data and the sampled negative data from S. Our framework can be easily applied for most recommendation models.

Optimizing the sampler model. Due to the discrete nature of candidate set G_u, we adopt the policy gradient method from reinforced learning to optimize model S. In other words, the optimization of S can be viewed as a reinforcement learning setting, where R gives a reward to action "selecting

sampled items for a user performed according to the policy probability $p_s(i|u)$." We can derive the policy gradient of L w.r.t. θ:

$$\begin{aligned}
\nabla_\theta L &= \sum_{u\in U}\left(\sum_{i\in\mathcal{X}_u}\nabla_\theta \log p_s(i|u;\theta) + N_u E_{p_s}[e_{ui}]\right)\\
&= \sum_{u\in U}\left(\sum_{i\in\mathcal{X}_u}\nabla_\theta \log p_s(i|u;\theta) + N_u \sum_{i\in I}\nabla_\theta p_s(i|u;\theta)e_{ui}\right)\\
&= \sum_{u\in U}\left(\sum_{i\in\mathcal{X}_u}\nabla_\theta \log p_s(i|u;\theta) + N_u E_{p_s}[\nabla_\theta \log p_s(i|u;\theta)e_{ui}]\right)\\
&\approx \sum_{u\in U}\left(\sum_{i\in\mathcal{X}_u}\nabla_\theta \log p_s(i|u;\theta) + \sum_{\alpha\in G_u, G_u\sim S}\nabla_\theta \log p_s(\alpha|u)e_{u\alpha}\right),
\end{aligned} \tag{14}$$

where $e_{ui} = (1 - x_{ui})\log(1 - f_\varphi(u,i))$ denotes the negative reward for the sampled negative items. In addition, we perform a sampling approximation in the last step in which item α is from the sampled item set G_u.

3.3 Collaborative Sampler with Integrated Framework

We further integrate the aforementioned collaborative sampler into our integrated framework. When the collaborative sampler meets the integrated framework, the collaborative signals can be well exploited. However, the training of S will suffer from inefficiency due to the heavy computation of the sampling probability and gradients. Thus, we further mitigate this efficiency problem by disassembling the sampling probability with different information paths. We can re-write the sampling probability of the instance in our collaborative sampler as

$$\begin{aligned}
p_s(i|u;\theta) = \rho_{ui} &= \sum_{k\in U}\mathbf{H}_{u,k}\frac{1}{m} + \mathbf{H}_{u,n+i}\\
&= \frac{1-c_1}{1-c_1c_2}\frac{1}{m} + \sum_{\mathcal{P}\in\mathcal{A}_{ui}}F(\mathcal{P})\\
&\equiv p_0 + p_{ARW}(\mathcal{P}, i|u;\theta),
\end{aligned} \tag{15}$$

where $F(\mathcal{P})$ denotes the path strength defined on Equation (8) and can be considered as the sampled probability of the path from the aforementioned ARW. Thus, we transfer $\nabla_\theta L$ (Equation (14)) into

$$\begin{aligned}
\nabla_\theta L &= \sum_{u\in U}\left(\sum_{i\in\mathcal{X}_u}\nabla_\theta \log p_s(i|u;\theta) + N_u\sum_{i\in I}\nabla_\theta p_s(i|u;\theta)e_{ui}\right)\\
&= \sum_{u\in U}\left(\sum_{i\in\mathcal{X}_u}\frac{\sum_{\mathcal{P}\in\mathcal{A}_{ui}}\nabla_\theta p_{ARW}(\mathcal{P}, i|u;\theta)}{p_s(i|u;\theta)}\right.\\
&\quad \left. + N_u\sum_{i\in I}\sum_{\mathcal{P}\in\mathcal{A}_{ui}}\nabla_\theta p_{ARW}(\mathcal{P}, i|u;\theta)e_{ui}\right)\\
&= \sum_{u\in U}\left(\sum_{i\in\mathcal{X}_u}\frac{E_{\mathcal{P},j\sim p_{ARW}}[\mathbf{I}[j=i]\nabla_\theta \log p_{ARW}(\mathcal{P}, i|u;\theta)]}{p_0 + E_{\mathcal{P},j\sim p_{ARW}}[\mathbf{I}[j=i]]}\right.\\
&\quad \left. + N_u E_{\mathcal{P},i\sim p_s}[\log\nabla_\theta p_{ARW}(\mathcal{P}, i|u;\theta)e_{ui}]\right),
\end{aligned} \tag{16}$$

ALGORITHM 1: Learning a recommendation model with CoSam

```
Initialize parameters φ, θ randomly;
while not converge do
    for each user u do
        Sample a set of items G_u and paths S_u with ARW.
    end for
    Calculate gradients ∇_φ L w.r.t. parameters φ of R based on G_u (Equation (13)).
    Update φ based on ∇_φ L.
    Calculate policy gradients ∇_θ L w.r.t. parameters θ of S based on S_u and G_u (Equation (17)).
    Update θ based on ∇_θ L.
end while
```

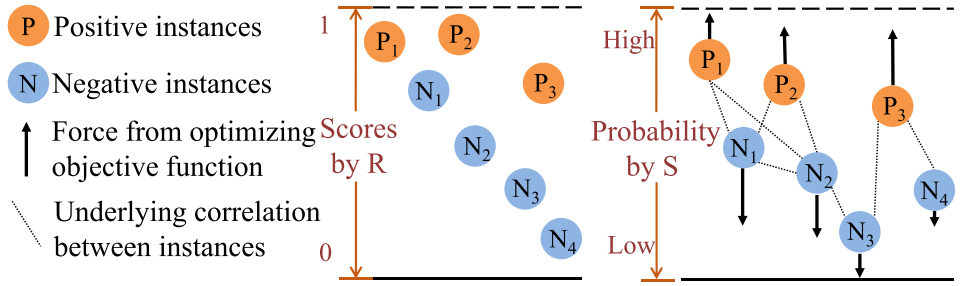


Fig. 4. Illustration of how CoSam achieves sampling informatively and collaboratively.

where $\mathbf{I}[.]$ denotes indicator function. The gradient can be approximately estimated based on the sampled paths for each user ($\mathcal{P} \in \mathcal{S}_u$) in a mini-batch:

$$\nabla_\theta L \approx \sum_{u \in U} \Bigg(\sum_{i \in \mathcal{X}_u} \frac{\sum_{\mathcal{P}, j \in \mathcal{S}_u} [\mathbf{I}[j = i] \nabla_\theta \log p_{RAW}(\mathcal{P}, i|u; \theta)]}{N_u p_0 + \sum_{\mathcal{P}, j \in \mathcal{S}_u} [\mathbf{I}[j = i]]} + \sum_{\mathcal{P}, j \in \mathcal{S}_u} [\log \nabla_\theta p_{RAW}(\mathcal{P}, i|u; \theta) e_{ui}] \Bigg). \tag{17}$$

The learning algorithm of CoSam is presented in Algorithm 1.

3.4 Discussion

3.4.1 Gradient Analysis. Sampling informatively. How do we understand the gradient of L w.r.t. parameters θ of S? Let us draw an analogy with the floating balls in the water as illustrated in Figure 4 for a better description. The first term in Equation (14) will give a force to push up these positive balls. Due to the correlations between the instances, which can be analogies with elastic links lines between the balls, the negative instances that have strong correlations with positive instances will be pulled up due to the force from the lines. This effect drives S to over-sample "positive-like" instances, which can reduce the sampling variance and improve convergence.

Sampling collaboratively. However, from the second term in Equation (14), the action of sampling a negative item will be punished with a negative reward e_{ui}. We observe an interesting phenomenon that when the sampled negative item is more hard to be differentiated by model R, the reward will be more negative as illustrated in Figure 4. This can be explained by the collaboration effect between S and R. Note that the negative items shall be differentiated either by model S or model R. If R fails to differentiate a negative item, it naturally turns to S for help and gives a signal to decrease the probability of selecting this item into the sampled item set. The reward guides S to flow to regions where S and R can well fit the data collaboratively. Moreover, if the sampling probability

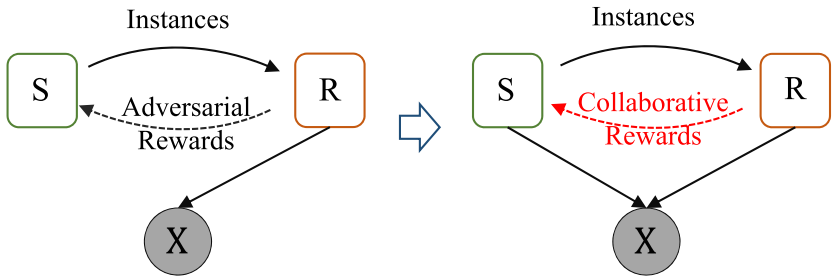


Fig. 5. Illustration of the previous adversarial framework (left) and our integrated framework (right).

of the item decreases with the effect of the reward, it suggests that the item can be differentiated by S and R's costly effort on this item will be spared; otherwise, S will maintain relatively high sampling probability so that R can focus on this "highly difficult" instance.

3.4.2 Complexity Analysis. The time cost on training CoSam consists of three parts. First, on sampling, it requires $O(nl_r\bar{N}_u)$ time to sample instances and paths, where $\bar{N}_u$ denotes the average number of sampled instances over all users. Second, the model R can be updated with the sampled instances. This part depends on the selected recommendation model, and its complexity is also linear to the number of sampled instances $O(n\bar{N}_u)$. Third, when updating model S, we just calculate rewards and estimate policy gradients from the sampled instances. The time for this part is also $O(n\bar{N}_u l_r)$. In practice, as in recent works [5], we usually let $\bar{N}_u$ be several times (e.g., five) as large as the number of the positive items of user u and thus have $n\bar{N}_u = 5|X^+|$, where $|X^+|$ denotes the number of observed positive data in the system. Overall, due to the sparsity of the recommendation data, our CoSam is efficient and can scale up to a large dataset.

3.4.3 Linking to Existing Methods. Adversarial framework. We describe two key differences of our integrated framework from an adversarial framework. First, as illustrated in Figure 5, in our integrated framework, S and R collaboratively contribute to the prediction, whereas the prediction in adversarial framework just depends on R. Second, both frameworks are trained in a reinforced manner. However, S in our framework is trained for collaboration with R, whereas S in the adversarial framework is trained for competition with R.

Two-stage recommendation strategy. Another similar work is the two-stage recommendation strategy [1, 7, 32, 58]. It contains a recalling model to retrieve "positive-like" candidate items from a global item pool and a re-ranking model to finely select recommendations from the candidate item set. This work is different from our integrated framework because the the two-stage recommendation strategy is not a mechanism for training a recommendation model. Instead, the two-stage recommendation strategy is an operation to accelerate retrieval from a pre-trained recommendation model by reducing the scale of the ranked items. Further, from a technical view, the intermediate item set in our framework is a "soft" probabilistic variable generated from the sampled model, whereas the intermediate item set in the two-stage recommendation strategy is a "hard" constant selected by the retrieval model.

4 RELATED WORK

Recommendation using implicit feedback data. Recent years have seen a surge of research on recommendation techniques using implicit feedback [24, 26, 37]. To handle the massive volume of the unobserved negative data, two strategies have been proposed in previous works. The first employs memorization strategies to accelerate learning on the whole data. For example, ALS and eALS [2, 20, 22] have been proposed to memorize some specific intermediate variables to avoid massive

repeated computations. However, this type of method is just suitable for the recommendation model with a K-separable property [2] and L2 loss function. The other type includes sampler-based methods, where the recommendation model is trained on the reduced sampling instances with a stochastic gradient descent optimizer. This method has been widely applied in many recommendation models, including classic matrix factorization [22, 25], pair-wised models (e.g., BPR [36], NCR[39]) and sophisticated neural network–based methods (e.g., CDL [41], DNN [48], NCF [19], xDeepFM [30], ConvNCF [18], NGCF [44], lightGCN[17], and Mult-VAE [31]).

Sampler for recommendation. The most popular sampling strategy is to sample negative data with equal probability (uniformly). Although efficient, the optimizer usually samples uninformative training instances [35, 50, 51], which makes limited contribution on updating. The model cannot be well trained, and the final recommendation performance will degrade. To deal with this problem, three types of sampling methods have been proposed in recent literature. The first is heuristic-based strategies, which pre-define a sampling rule to over-sample informative instances [21, 35, 50, 53, 55]. For example, some works [21, 50, 52] propose to sample instances based on global item popularity, whereas others [35, 51, 52, 54] propose to draw instances based on item ranking position. However, these methods require rich human experience and also lack flexibility. Another type is the model-based sampler [10, 13, 34, 42, 43], which defines a sampler model and the sampling probability that will adaptively evolve by optimizing an adversarial objective. However, as analyzed in Section 2, this type of method will suffer from inefficiency and biases. In addition, some works leverage auxiliary information to sample informative training instances. For example, some works [5, 45, 55] claim to over-sample the items that have been consumed by social friends, whereas others [9–11] leverage exposure data to find a reliable negative instance. Wang et al. [46] leverage a knowledge graph to enhance negative sampling. However, such auxiliary information may not be available in many recommendation systems.

Random walk in recommendation. The random walk strategy has been widely applied in recommendation. Jamali and Ester [23] perform random walk along the social network to search relevant users who preference similar to the target user for better rating prediction. Christoffel et al. [6] exploit random walk to obtain diverse recommendation. Vahedian et al. [40] further extend the work of Christoffel et al. [6] in a heterogenous information network to generate valuable meta-paths. Christoffel et al. [6] further perform random walk to complete an implicit feedback matrix for mitigating the data sparsity problem. Similarly, Yu et al. [51] employ random walk to find more positive instances. We remark that these works adopt static (uniform or pre-defined) transfer probability in their random walk strategy. In addition, these random walks are not designed for sampling informative negative instances.

To the best of our knowledge, only one recent work on SamWalker [5] employs adaptive random walk to sample negative instances for better training of the recommendation model. The random walk in our CoSam differs from SamWalker in that (1) the random walk in SamWalker is performed on the social network, whereas in CoSam it is on the user-item interaction graph, and (2) the transfer probability in SamWalker is learned from users' exposure, whereas in CoSam it is learned with the rewards from the recommendation model.

5 EXPERIMENTS

In this section, we conduct experiments to evaluate the performance of our CoSam. Our experiments are intended to address the following research questions:

(RQ1) How does CoSam compare with existing sampling methods?
(RQ2) Does CoSam boost recommendation performance?
(RQ3) How do different components in CoSam (i.e., the collaborative sampling model and integrated framework) affect CoSam's performance?

Table 2. Statistics of Four Datasets

Datasets	LastFM	MovieLens	Ciao	Epinions
Number of users	1,892	6,040	5,298	21,290
Number of items	4,476	3,678	19,264	33,992
Number of positive feedback	52,627	1,000,177	138,723	333,658
Density of positive feedback	0.62%	4.50%	0.14%	0.05%

5.1 Experimental Settings

Datasets and parameters. Four well-known recommendation datasets (LastFM,[1] MovieLens-1M,[2] Ciao,[3] and Epinions[4]) are used in our experiments. These datasets contain users' feedback on items. The dataset statistics are presented in Table 2. We preprocess the datasets so that all items have at least three interactions and "binarize" the user's feedback into implicit feedback as in other works [5, 47]. In other words, as long as there exist some user-item interactions (ratings or clicks), the corresponding implicit feedback is assigned a value of 1. Grid search and fivefold cross validation are used to find the best hyper-parameters and test the results. In our CoSam, as in recent works [5, 50], we set $N_u = 5|\mathcal{X}_u|$ and test $c1, c2$ of [0.4,0.6,0.8,1]. We also optimize the model with mini-batch Adam [27] and test the learning rate of [0.001,0.01,0.1]. The setting of compared methods refers to related works or is tested in our experiments. All experiments are conducted on a server with two Intel E5-2620 CPUs, two NVIDIA Titan-X GPUs, and 256 G of RAM. We will share our source code at GitHub[5] when the work is published.

Evaluation metrics. We adopt the following metrics to evaluate recommendation performance:

- *Recall@K (Rec@K)*: This metric quantifies the fraction of consumed items that are in the top-K ranking list sorted by their estimated rankings. For each user u, we define $Rec(u)$ as the set of recommended items in top-K and $Con(u)$ as the set of consumed items in test data for user u. Then we have

$$Recall@K = \frac{1}{|U|} \sum_{u \in U} \frac{|Rec(u) \cap Con(u)|}{|Con(u)|}. \tag{18}$$

- *Precision@K (Pre@K)*: This measures the fraction of the top-K items that are indeed consumed by the user:

$$Precision@K = \frac{1}{|U|} \sum_{u \in U} \frac{|Rec(u) \cap Con(u)|}{|Rec(u)|}. \tag{19}$$

- *Normalized Discounted Cumulative Gain*: This is widely used in information retrieval, and it measures the quality of ranking through discounted importance based on positions. In recommendation, **Normalized Discounted Cumulative Gain (NDCG)** is computed as follows:

$$NDCG = \frac{1}{|U|} \sum_{u \in U} \frac{DCG_u}{IDCG_u}, \tag{20}$$

[1] https://grouplens.org/datasets/hetrec-2011/.
[2] https://grouplens.org/datasets/movielens/.
[3] https://www.cse.msu.edu/~tangjili/trust.
[4] http://www.trustlet.org/epinions.
[5] https://github.com/jiawei-chen/CoSam.

Table 3. The Performance Metrics of the Compared Samplers

Methods	LastFM				MovieLens-1M			
	Pre@5	Rec@5	NDCG	Time	Pre@5	Rec@5	NDCG	Time
Uniform	0.0954	0.0859	0.3455	18.7	0.3603	0.0790	0.5669	280
Pop	0.0978	0.0881	0.3513	21.6	0.3633	0.0811	0.5692	329
Adp	0.0986	0.0881	0.3448	24.1	0.3635	0.0802	0.5667	315
Adv	0.1002	0.0897	0.3503	104	0.3691	0.0811	0.5689	1212
CoSam	**0.1302**	**0.1162**	**0.3838**	23.2	**0.3772**	**0.0837**	**0.5726**	672
Impv%	23.00%	29.53%	9.24%	-	2.17%	3.15%	0.59%	-
Methods	Ciao				Epinions			
	Pre@5	Rec@5	NDCG	Time	Pre@5	Rec@5	NDCG	Time
Uniform	0.0140	0.0122	0.1775	65.0	0.0085	0.0095	0.1557	156
Pop	0.0146	0.0119	0.1773	91.6	0.0092	0.0098	0.1565	264
Adp	0.0146	0.0122	0.1784	94.5	0.0088	0.0095	0.1559	348
Adv	0.0153	0.0139	0.1786	980	0.0094	0.0103	0.1567	8760
CoSam	**0.0185**	**0.0149**	**0.1807**	103	**0.0153**	**0.0168**	**0.1621**	420
Impv%	20.93%	7.48%	1.20%	-	61.72%	62.72%	3.45%	-

The boldface font denotes the winner in that column. The rows 'Impv' indicate the relative performance gain of CoSam over the best baseline. † indicates that the result of a paired difference test is significant at $p < 0.05$. We also present running time (second) for comparison.

where DCG_u is defined as follows and $IDCG_u$ is the ideal value of DCG_u coming from the best ranking:

$$DCG_u = \sum_{i \in Con(u)} \frac{1}{\log_2(rank_{ui} + 1)}, \tag{21}$$

where $rank_{ui}$ represents the rank of item i in the recommended list of user u. NDCG can be interpreted as the ease of finding all consumed items, as higher numbers indicate that the consumed items are higher in the list.

5.2 Comparison with Existing Samplers (RQ1)

To answer the first research question (RQ1), we compare our CoSam with other sampling strategies, including:

- *Uniformly (Uni)*: A widely adopted sampling strategy that samples items with equal probability.
- *Item-Popularity (Pop)* [50]: Samples items based on item popularity: $p(i|u) \propto \xi_i^\alpha$, where ξ_i denotes the popularity of item i and parameter α scales its effect.
- *Adaptive oversampling (Adp)* [35]: The adaptive sampling strategy to over-sample the "difficult" items for the MF-based recommendation model.
- *Adversarial sampler (Adv)*: The state-of-the-art model-based sampler with an adversarial sampler-recommender framework. Since the sampler models of Chen et al. [5] and Ding et al. [10] require other side information, we follow the work of Park and Chang [34] and Wang et al. [42] and model the sampler with matrix factorization.

In addition, for a fair comparison, we integrate these samplers into a common matrix factorization recommendation model. Their performance is reported in Table 3.

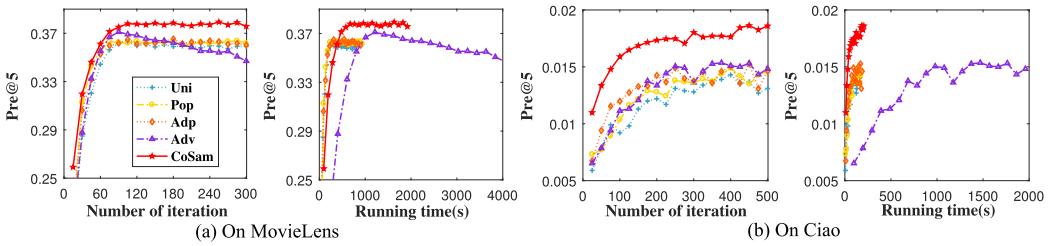


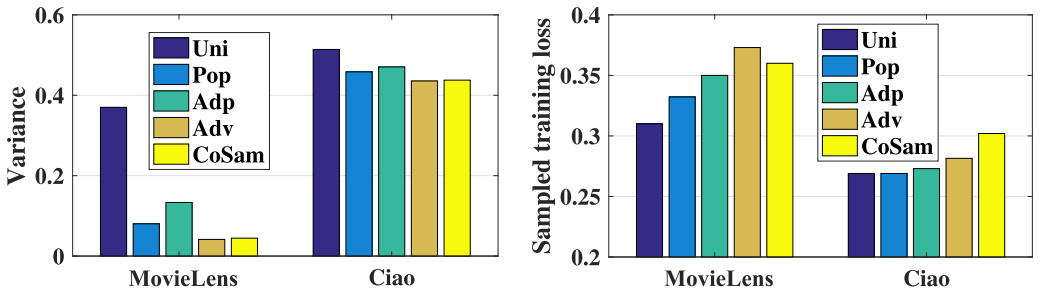
Fig. 6. Pre@5 for different sampling strategy versus the number of iterations and running time.

Performance comparison. Table 3 presents the recommendation performance of the compared sampler in terms of three evaluation metrics. The boldface font denotes the winner in that column. Overall, CoSam outperforms all compared methods on all datasets for all metrics. For the sake of clarity, the last row of Table 3 also shows the relative improvements achieved by CoSam over the best baselines. We have the following observations. First, the informative sampler will boost the recommendation performance. It can be seen from the better performance of Adv, Ada, and CoSam over Uni. Second, the model-based samplers (CoSam and Adv), which will evolve with the training, outperform these heuristic methods. This result is consistent with our intuition, since the heuristic samplers lack flexibility and usually fail to capture the varying informativeness of instances. Third, CoSam outperforms the compared methods by a large margin. The average improvement of CoSam over the best baselines on the four datasets is 26.9%, 25.7%, and 3.0% in terms of Pre@5, Rec@5, and NDCG, respectively. Especially in the large sparse dataset Epinions, the improvements are quite impressive—over 50% in both Pre@5 and Rec@5. This can be attributed to the collaborative sampler model and integrated sampler-recommender framework. We will conduct ablation studies in Section 5.4 to analyze the effect of these components.

Comparison in terms of datasets. The gap between CoSam and the best baseline is generally larger on Epinions and smaller on MovieLens. This phenomenon can be explained by the different levels of the sparsity of the datasets. Drawing informative instances from a highly sparse dataset is more challenging, and existing samplers perform worse in this dataset. Another phenomenon that can be observed is that scores of metrics are relatively higher on MovieLens and quite lower on Epinions and Ciao. The reason can also be attributed to the sparsity. In the typical sparse dataset Epinions, most users just have a few positive items. It is difficult for a recommendation model to retrieve such limited positive recommendations, as they are buried in a large pile of negative ones.

Running time. Table 3 also presents the training time cost of the compared samplers, and Figure 6 depicts Pre@5 (Y axis) vs. the number of iterations or running time (X axis) of these methods on the two typical datasets. The powerful competitor Adv in accuracy spends much more time on sampling. Especially in the largest dataset Epinions, the recommendation method with Adv requires 2.4 hours for training, whereas CoSam just requires 7 minutes. A 20-time acceleration will be achieved by our CoSam. The speedup can be attributed to our collaborative sampler, which avoids time-consuming traversal of all items. This way, CoSam has time complexity similar to these efficient heuristical methods, which aim at fast learning but sacrifice certain flexibility. Their actual running times are also in the same magnitude.

Variance and training loss. To show the stability and informativeness of samplers, we further conduct specific experiments on empirically comparing the variance of the estimated gradient $\nabla_L \varphi$ and average training loss on sampled instances. To do so, we train the models for 50,100,150 epochs and then generate a mini-batch with various sampling strategies to estimate normalized gradients and training loss. We repeat this process 1,000 times, and calculate the variance of the



(a) Variance comparison (b) Training loss comparison

Fig. 7. Variance of gradients and sampled training loss using different sampling strategies.

gradients and average training loss for different sampling strategies. The final results are averaged over various parameters and are presented in Figure 7. The sampler value of variance suggests that the model will be more stable, whereas the larger values of the sampled training loss indicate that the sampled instances will be more informative. Figure 7 shows that our CoSam outperforms these heuristic samplers and achieves similar performance with Adv. We note that Adv will sacrifice training efficiency. In addition, Adv will increase the risk of over-fitting as illustrated in Figure 6.

5.3 Effect of CoSam (RQ2)

To answer the second research question (RQ2), we apply our CoSam sampling methods to two typical recommendation models (MF and NCF) and explore how CoSam boosts recommendation performance. We compare the improved recommendation methods—MF-CoSam and NCF-CoSam—with the following recommendation methods:

- *MF* [22, 25, 33]: The classic matrix factorization model for implicit feedback data with Bernoulli likelihood and a uniform sampler.
- *BPR* [36]: The classic pairwise method for recommendation, coupled with matrix factorization.
- *NCF*: [19]: The advanced recommendation method modeling user-item interactions with a neural network. We refer to the work of He et al. [19] and train the model with a uniform sampler.
- *IRGAN* [42]: The method that employs a generative adversarial model to capture users' preference over the items. Here we just report the results of IRGAN on the small dataset LastFM and MovieLens due to its massive time cost on large datasets.
- *NGCF* [44]: The state-of-the-art graph-based recommendation method, which encodes users and items based on a neural graph network [16] on the user-item interaction graph.

Figure 8 presents the recommendation performance of the compared methods. The improvement from CoSam is apparent: adopting CoSam consistently boosts the recommendation performance. Especially on the large dataset Epinions, the improvements are over 30% achieved by MF-CoSam (or NCF-CoSam) compared with MF (or NCF). Further, we observe that these sampler-enhanced methods can beat state-of-the-art recommendation methods, even if we just employ a relatively simple recommendation model (i.e., MF). This result validates the importance of the sampling method, which makes a nonnegligible contribution on recommendation performance. In fact, the recommendation model has been well exploited by substantial recent works, whereas the sampler usually becomes the effect bottleneck of the recommendation system.

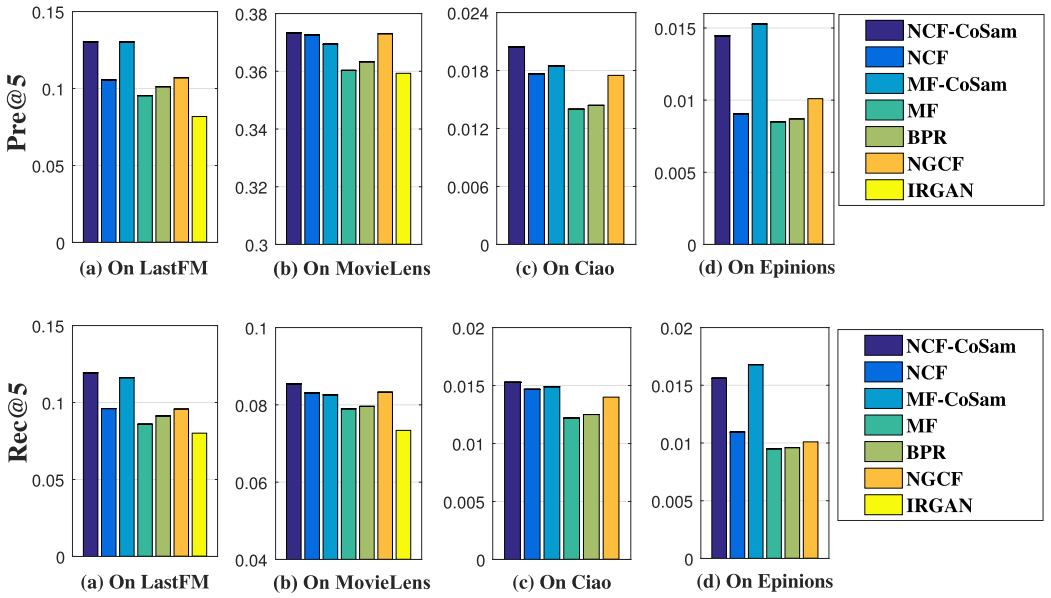


Fig. 8. Performance of different recommendation methods.

Table 4. The Performance and Running Time of CoSam and Its Variants

Methods	LastFM				MovieLens-1M			
	Pre@5	Rec@5	NDCG	Time	Pre@5	Rec@5	NDCG	Time
Sam-MF-Adv (Adv)	0.1002	0.0897	0.3503	104	0.3691	0.0811	0.5689	1414
Sam-MF-Int	0.1073	0.0957	0.3579	116	0.4043	0.0953	0.5962	1484
Sam-Co-Adv	0.0994	0.0892	0.3454	19	0.3643	0.0805	0.5676	630
Sam-Co-Int(CoSam)	0.1302	0.1162	0.3838	23	0.3772	0.0837	0.5726	672
Methods	Ciao				Epinions			
	Pre@5	Rec@5	NDCG	Time	Pre@5	Rec@5	NDCG	Time
Sam-MF-Adv (Adv)	0.0153	0.0139	0.1786	980	0.0094	0.0103	0.1567	8760
Sam-MF-Int	0.0160	0.0131	0.1797	977	0.0125	0.0134	0.1545	9984
Sam-Co-Adv	0.0154	0.0138	0.1788	102	0.0091	0.0098	0.1562	322
Sam-Co-Int(CoSam)	0.0185	0.0149	0.1807	103	0.0153	0.0168	0.1621	420

5.4 Ablation Study (RQ3)

We further design a detailed ablation study to answer the third research question (RQ3). In other words, we remove different components at a time and compare CoSam with its three special cases: Sam-MF-Int, Sam-Co-Adv, and Sam-MF-Adv. Here we denote Sam-X-Y as the sampler with the "X" sampler model and the "Y" sampler-recommender framework. Here "MF" denotes the matrix factorization model, and "Co" denotes the collaborative model; and "Adv" denotes the adversarial framework, and "Int" denotes the integrated framework. CoSam can be denoted as Sam-Co-Int, and Adv can be denoted as Sam-MF-Adv. We integrate these methods into the benchmark MF recommendation model to test their performance. The recommendation performance and training running time are presented in Table 4.

Table 5. Characteristics of Sam-MF-Int and Its Variants

Methods	Training Criterion	Prediction with S?	Prediction with R?
Sam-MF-Int	Integrated	√	√
Sam-MF-Int-S	Integrated	√	\
Sam-MF-Int-R	Integrated	\	√
Sam-MF-Adv-U	Adversarial	√	√
Sam-MF-Adv	Adversarial	\	√

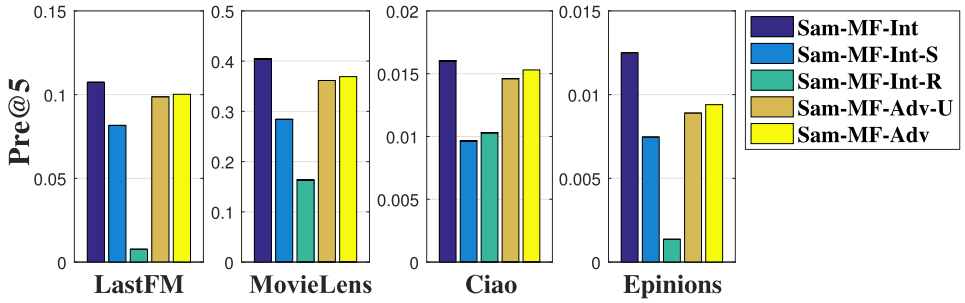


Fig. 9. Performance of Sam-MF-Int and its variants.

We observe that the methods with the integrated sampler-recommender framework (Sam-Co-Int and Sam-MF-Int) outperform their variants with the adversarial framework (Sam-Co-Adv and Sam-MF-Adv). This is because the ability of the sampler model has been better exploited, and the sampling biases have been refined in the collaborative framework. For the sampler model comparison, generally the collaborative sampler model outperforms the MF-based sampler in the collaborative framework, whereas it performs worse in the adversarial framework. This interesting phenomenon can be explained as follows. The collaborative sampler, which encodes rich interaction information into sampling, will produce more informative instances. Naturally, it will achieve better performance under an integrated framework. However, when it is applied to the adversarial framework, this informative but highly biased sampler will heavily deteriorate the performance and offset the advantage from informativeness. It is noteworthy that the collaborative sampler model performs much more efficiently than the MF-based sampler model. Thus, the collaborative sampler will be more suitable for the modern recommendation system with large item space.

Effect of integrated recommendation. Another interesting ablation experiment has been conducted to show the effect of integrating S and R into prediction for recommendation. In other words, we compare Sam-MF-Int with the following variants:

- *Sam-MF-Int-S and Sam-MF-Int-R*: The special cases of Sam-MF-Int, which trains S and R based on integrated framework but makes recommendation just based on S (Sam-MF-Int-S) or R (Sam-MF-Int-R).
- *Sam-MF-Adv-U*: The variant of Sam-MF-Adv, which trains S and R based on the adversarial framework but unifies S and R for recommendation as the integrated framework.

Their characteristics are presented in Table 5. Here we choose Sam-MF-Int instead of Sam-Co-Int (CoSam) for experiments to isolate the effect of the collaborative sampler model.

Figure 9 presents the performance of these compared methods. We observe that Sam-MF-Int consistently outperforms Sam-MF-Int-R and Sam-MF-Int-S, which trains S and R collaboratively but makes recommendations separately. This result validates the effectiveness of unifying S and R into prediction. The performance will degrade if we only consider one part. R will be biased to focus on these "difficult" instances, and its performance on other relatively "easy" instances will be deteriorated. Similarly, S is not trained to capture users' preference individually, and thus Sam-MF-Int-S performs inferiorly to some benchmark recommendation methods. Only when R meets S in recommendation will they will compensate each other and achieve impressive performance. However, we observe that unifying R and S into recommendation with an adversarial framework is ineffective. Sam-MF-Adv-U performs worse instead of better than Sam-MF-Adv. This can be attributed to the different rewards (as discussed in Section 3.4). In the adversarial framework, S is trained for competition instead of collaboration with R. Thus, S and R are not ready for collaborative prediction.

6 CONCLUSION

In this article, we present CoSam, a novel efficient and effective sampling method. It consists of (1) a collaborative sampler model that leverages collaborative signals into sampling and satisfies the desirable property of normalization, adaption, interaction information awareness, and computational efficiency, and (2) an integrated sampler-recommender framework to refine the sampling bias and to better exploit the ability of the sampler model. Our extensive experiments on four real-world datasets validate the superiority of the proposed collaborative sampler and integrated framework.

One interesting direction for future work is to leverage a sophisticated graph neural network in a sampler process to capture more complex affinity between users and items along the interaction graph. In addition, effectiveness and efficiency are dual objectives for sampling strategies. It will be interesting and challenging to design a more advanced sampler meeting both conditions. This kind of sampler is also beneficial to other tasks, such as graph embedding [14], social influence estimation [38], and unbalanced classification [8]. Another interesting direction is to explore a dynamic sampling strategy in an online recommender system, where user preferences will drift with time [29].

REFERENCES

[1] Nima Asadi and Jimmy Lin. 2012. Fast candidate generation for two-phase document ranking: Postings list intersection with Bloom filters. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 2419–2422.

[2] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*. 1341–1350.

[3] Jeffery P. Bray. 2008. *Consumer Behaviour Theory: Approaches and Models*. Discussion Paper. Bournemouth University.

[4] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and debias in recommender system: A survey and future directions. arXiv:2010.03240

[5] Jiawei Chen, Can Wang, Sheng Zhou, Shi, Yan Feng, and Chun Chen. 2019. SamWalker: Social recommendation with informative sampling strategy. In *Proceedings of the World Wide Web Conference*. ACM, New York, NY, 228–239.

[6] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. 2015. Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *Proceedings of the Conference on Recommender Systems*.

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, New York, NY, 191–198.

[8] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. 2015. Calibrating probability with undersampling for unbalanced classification. In *Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence*. IEEE, Los Alamitos, CA, 159–166.

[9] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An improved sampler for Bayesian personalized ranking by leveraging view data. In *Companion Proceedings of the Web Conference*. 13–14.
[10] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced negative sampling for recommendation with exposure data. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2230–2236.
[11] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. 2019. Sampler design for Bayesian personalized ranking by leveraging view data. *IEEE Transactions on Knowledge and Data Engineering* 33, 2 (2019), 667–681.
[12] Hande Dong, Jiawei Chen, Fuli Feng, Xiangnan He, Shuxian Bi, Zhaolin Ding, and Peng Cui. 2020. On the equivalence of decoupled graph convolution network and label propagation. arXiv:2010.12408
[13] Wenqi Fan, Tyler Derr, Yao Ma, Jianping Wang, Jiliang Tang, and Qing Li. 2019. Deep adversarial social recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1351–1357.
[14] Hongchang Gao and Heng Huang. 2018. Self-paced network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 1406–1415.
[15] 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.
[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
[17] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. arXiv:2002.02126
[18] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer product-based neural collaborative filtering. arXiv:1808.03912
[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. ACM, New York, NY, 173–182.
[20] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 549–558.
[21] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Stochastic inference for scalable probabilistic modeling of binary matrices. In *Proceedings of the International Conference on Machine Learning*. 379–387.
[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*. IEEE, Los Alamitos, CA, 263–272.
[23] Mohsen Jamali and Martin Ester. 2009. TrustWalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 397–406.
[24] Chen Jiawei, Feng Yan, Ester Martin, Zhou Sheng, Chen Chun, and Can Wang. 2018. Modeling users' exposure with social knowledge influence and consumption influence for recommendation. In *Proceedings of the 27th ACM International on Conference on Information and Knowledge Management*. ACM, New York, NY, 953–962.
[25] Christopher C. Johnson. 2014. Logistic matrix factorization for implicit feedback data. In *Advances in Neural Information Processing Systems* 27. 1–9.
[26] Diane Kelly. 2005. Implicit feedback: Using behavior to infer relevance. In *New Directions in Cognitive Information Retrieval*. Springer, 169–186.
[27] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980
[28] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
[29] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 304–312.
[30] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 1754–1763.
[31] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 689–698.
[32] Zheng Liu, Yu Xing, Jianxun Lian, Defu Lian, Ziyao Li, and Xing Xie. 2019. A novel user representation paradigm for making personalized candidate retrieval. arXiv:1907.06323
[33] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the 2008 8th IEEE International Conference on Data Mining (ICDM '08)*. 502–511. DOI: https://doi.org/10.1109/ICDM.2008.16

[34] Dae Hoon Park and Yi Chang. 2019. Adversarial sampling and training for semi-supervised information retrieval. In *Proceedings of the World Wide Web Conference*. ACM, New York, NY, 1443–1453.

[35] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, 273–282.

[36] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[37] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 501–509.

[38] Qihao Shi, Can Wang, Deshi Ye, Jiawei Chen, Yan Feng, and Chun Chen. 2019. Adaptive influence blocking: Minimizing the negative spread by observation-based policies. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE '19)*. IEEE, Los Alamitos, CA, 1502–1513.

[39] Bo Song, Xin Yang, Yi Cao, and Congfu Xu. 2018. Neural collaborative ranking. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 1353–1362.

[40] Fatemeh Vahedian, D. Robin Burke, and Bamshad Mobasher. 2017. Weighted random walk sampling for multi-relational recommendation. In *Proceedings of the 25th Conference on User Modeling, Adaptation, and Personalization (UMAP'17)*. 230–237.

[41] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 1235–1244.

[42] 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, New York, NY, 515–524.

[43] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2467–2475.

[44] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. 1905.08108

[45] Xin Wang, Wei Lu, Martin Ester, Can Wang, and Chun Chen. 2016. Social recommendation with strong and weak ties. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, New York, NY, 5–14.

[46] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *Proceedings of the 2020 Web Conference*. 99–109.

[47] Lin Xiao, Zhang Min, Zhang Yongfeng, Liu Yiqun, and Ma Shaoping. 2017. Learning and transferring social and item visibilities for personalized recommendation. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*. ACM, New York, NY, 337–346.

[48] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 3203–3209.

[49] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 974–983.

[50] 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*. 363–371.

[51] Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang. 2018. WalkRanker: A unified pairwise ranking model with multiple relations for item recommendation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*.

[52] Fajie Yuan, Guibing Guo, Joemon M. Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. LambdaFM: Learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 227–236.

[53] Fajie Yuan, Guibing Guo, Joemon M. Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2017. BoostFM: Boosted factorization machines for top-n feature-based recommendation. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. ACM, New York, NY, 45–54.

[54] 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, New York, NY, 785–788.

[55] Tong Zhao, Julian McAuley, and Irwin King. 2014. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 261–270.

[56] Denny Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*. 321–328.

[57] Sheng Zhou, Xin Wang, Jiajun Bu, Martin Ester, Pinggang Yu, Jiawei Chen, Qihao Shi, and Can Wang. 2020. DGE: Deep generative network embedding based on commonality and individuality. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 6949–6956.

[58] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 1079–1088.

Received June 2020; revised December 2020; accepted February 2021