

CoSam: An Efficient Collaborative Adaptive Sampler for Recommendation

JIAWEI CHEN, CHENGQUAN JIANG, CAN WANG[†], SHENG ZHOU, YAN FENG, and CHUN CHEN, Zhejiang University, China
 MARTIN ESTER, Simon Fraser University, Canada
 XIANGNAN HE, University of Science and Technology of China, 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 model’s convergency, 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 an efficient and effective collaborative sampling method CoSam, which 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. 2020. CoSam: An Efficient Collaborative Adaptive Sampler for Recommendation. *ACM Transactions on Information Systems* 37, 4, Article 111 (August 2020), 21 pages. <https://doi.org/10.1145/1122445.1122456>

1 INTRODUCTION

Recent research attention on recommendation is increasingly shifting from explicit feedback data (a.k.a. rating data) towards implicit feedback data. Implicit feedback data is often a natural byproduct of users’ behavior, and thus is more prevalent, inexpensive and scalable. However, recommendation from implicit feedback is very challenging due to the lack of negative feedback data. While the unobserved feedback data can be treated as a

[†]Corresponding author: wcan@zju.edu.cn.

Authors’ addresses: Jiawei Chen, sleepyhunt@zju.edu.cn; Chengquan Jiang, imjcqt@gmail.com; Can Wang[†], wcan@zju.edu.cn; Sheng Zhou, zhousheng_zju@zju.edu.cn; Yan Feng, fengyan@zju.edu.cn; Chun Chen, chenc@cs.zju.edu.cn, Zhejiang University, 38 ZheDa Road, Hangzhou, China; Martin Ester, Simon Fraser University, 8888 University Drive, Burnaby, Canada, ester@cs.sfu.ca; Xiangnan He, University of Science and Technology of China, 96 JinZhai Road, Hefei, China, xiangnanhe@gmail.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.
 1046-8188/2020/8-ART111 \$15.00
<https://doi.org/10.1145/1122445.1122456>

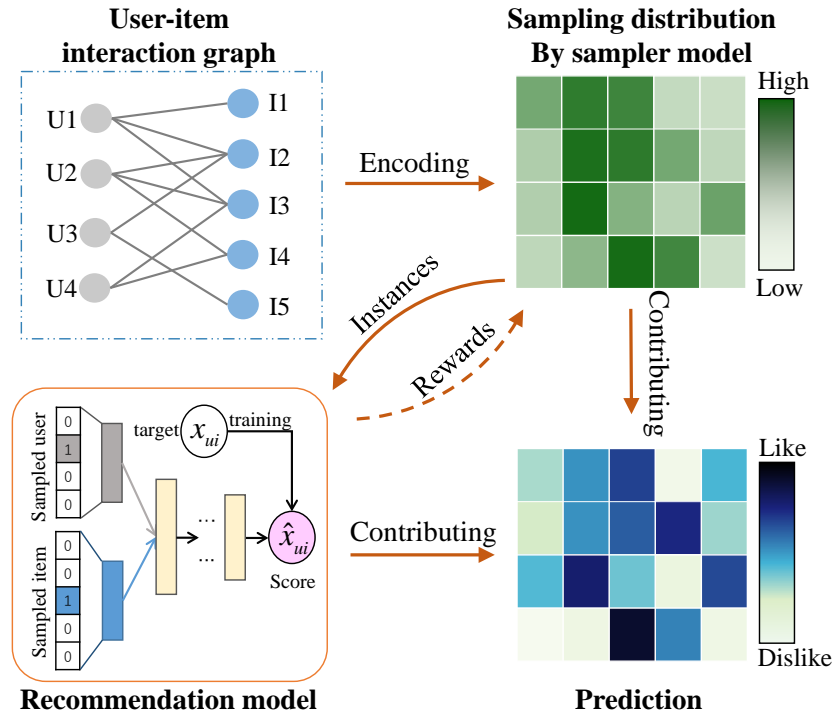


Fig. 1. Illustration of CoSam, which leverages interaction graph into sampling and integrates sampling distribution (model) into prediction. The user-item interaction graph is constructed from user-item interaction information, where we give links for the user-item pair with positive feedback.

source of negative signal, its large scale incurs expensive time cost in learning a recommendation model. Existing methods mainly rely on stochastic gradient descent optimizer (SGD) 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.

While there are rich literatures on sampling strategies for better recommendation, 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 [31, 43, 45].
- **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 samplers is that the “difficult” instances, which can be defined as the items with higher popularity [43] or higher ranking position [31], 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 [8, 30, 36]. Existing model-based samplers mainly adopt adversarial learning framework. With adversarial training between the sampler model and the recommender model, the sampler will learn to fit

users' preference distribution and thus adaptively generate positive-like “difficult” instances. While this type of samplers have been proved effective, they tend to be very slow in large dataset. The reason is that the sampling distribution will dynamic evolve with the training proceeding, 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 [4, 8]. However, these auxiliary information may not be available in many recommender systems.

In a word, 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:

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

In this paper, we propose a novel model-based adaptive sampler named **Collaborative Sampler** (CoSam). Corresponding to two problems, CoSam has two improvements and “collaborative” has two-fold meanings. First, to tackle the 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 the user U1 and item I4 (e.g. U1-I2-U2-I4, U1-I3-U4-I4). It reflects 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 the problem (P2), we propose a novel integrated sampler-recommender framework to replace 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 can not 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 an 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 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 paper 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 paper 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 from 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 a $n \times m$ binary matrix \mathbf{X} with entries $x_{ui} \in \{0, 1\}$ denoting whether or not the user u has interacted the item i . Also, 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 follow: learning user's preference from the available implicit feedback and recommending items that are most likely to be interacted by the user.

2.2 Adversarial sampler-recommender framework

The adversarial sampler-recommender framework contains:

a sampler model (S): $G_u \sim \text{Multinomial}(N_u, p_s(i|u; \theta))$, 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" instances for recommender. We remark that some works [8] will remove the sampled positive instances from G_u .

a recommender model (R): which captures 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 \mathcal{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 Bernoulli classifier as an example for further derivation and it can be easily replaced by other functions (e.g. Gaussian likelihood [29], Bayesian Personalized Ranking [32]). Formally, we have:

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

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

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

where $E_{G_u \sim S}[\cdot]$ stands for $E_{G_u \sim \text{Multinomial}(N_u, p_s(i|u; \theta))}[\cdot]$. 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 kind of methods: (1)

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. (2) The adversarial uneven sampler will create biases on the recommendation model, leading to non-optimal results. It can be understood from the expected objective function of the recommender model in the adversarial framework:

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

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 [10, 11]. 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 paper, we are interested in, and propose a solution to two related problems:

- (1) A new sampler model support efficient and effective sampling.
- (2) A new sampler-recommender framework to mitigate biases caused by 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 the interaction graph structure, which contains rich correlations information among users and items. Inspired by the success of the graph convolutional network model [23, 37, 42] 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 a one-hot vector in which i -th element is equal to 1. The model will propagation users and items information along the graph to refine users' sampling distribution and naturally inject useful correlation information into sampling distribution. Concretely, as shown in Figure 2, in each layer, each node in the graph collect information from the connected neighbors and reconstruct their distribution or labels as follows:

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

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

The parameters c_1, c_2 ($0 \leq c_1, c_2 \leq 1$) specify the relative contributions from the neighbors and the initial vectors. $w_{ui}^{(1)}, w_{iu}^{(2)}$ are learned parameters, which balances the heterogenous influence from different neighborhoods and meets $\sum_{i \in I, i \in N_i} w_{ui}^{(1)} = 1, \sum_{u \in \mathcal{U}, u \in N_i} w_{iu}^{(2)} = 1, w_{ui}^{(1)} > 0, w_{iu}^{(2)} > 0$. Overall, CoSam models sampling probability with an

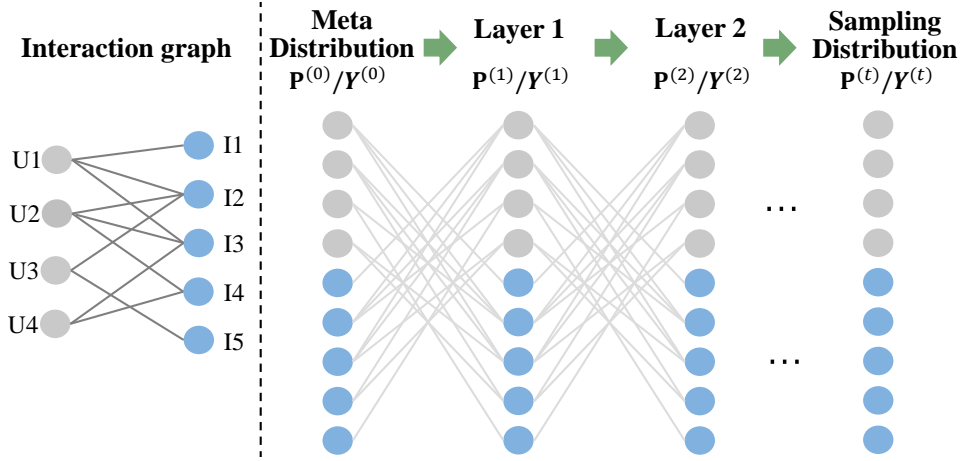


Fig. 2. Illustration of the collaborative sampler model.

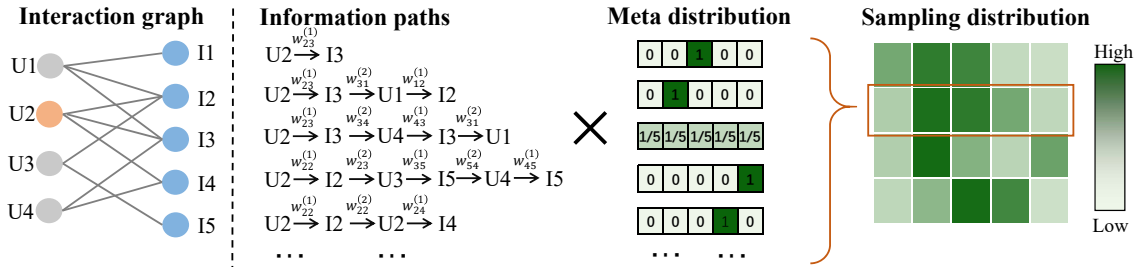


Fig. 3. The constructed sampling distribution of CoSam can be considered as a mixture of meta distributions, where the mixing coefficients are controlled by the information paths and the learned path strength.

interactive graph-aware function g parameterized by edge weights w ($w^{(1)}, w^{(2)}$), to which eq. (4),(5) converges:

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

where we collect sampling distribution ρ_u for each user as matrix \mathbf{P} and collect labels \mathbf{y}_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; 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. The constructed sampling model must have a valid probability distribution over the item space, i.e. for each user u , we have $\sum_{i \in I} \rho_{ui} = 1$.

PROOF. Let mark the adjacency matrix of the interactive graph as $\mathbf{W} = \begin{bmatrix} \mathbf{0} & \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} & \mathbf{0} \end{bmatrix}$. Here we mark user nodes in the interaction graph with id 1 to n , and item nodes with id $n+1$ to $n+m$. Let \mathbf{e} denotes the column

vector of $n + m$ ones. we have: $(I - CW)\mathbf{e} = (I - C)\mathbf{e}$. Note that $(I - CW)^{-1}(I - CW)\mathbf{e} = \mathbf{e}$. Thus, we can get $(I - cW)^{-1}(I - C)\mathbf{e} = \mathbf{e}$. Then, we have $\begin{bmatrix} \mathbf{P}^{(t)} \\ \mathbf{Y}^{(t)} \end{bmatrix} \mathbf{e} = (I - cW)^{-1}(I - C) \begin{bmatrix} \mathbf{P}^{(0)} \\ \mathbf{Y}^{(0)} \end{bmatrix} \mathbf{e} = \mathbf{e}$. \square

Interaction information awareness. As we can see from eq.(6), the matrix $\mathbf{H} = (I - CW)^{-1}(I - C)$ is a graph or diffusion kernel [49], 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 indicator distribution), where the mixing coefficients are specified with the adaptive interaction graph proximity, which will evolve with training process going on. That is, we have the following formula of the sampling distribution of user u :

$$\rho_u = \sum_{k \in U} \mathbf{H}_{u,k} \mathcal{U} + \sum_{k \in I} \mathbf{H}_{u,n+k} \mathcal{Z}_k \quad (7)$$

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

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

where \mathcal{P} denotes a path instance on the graph and \mathcal{A}_{ij} denotes the set of path \mathcal{P} between node i and j . $F(\mathcal{P})$ is the product of edge weights on the path, which can be defined as 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 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:

Adaptive random walk strategy (ARW for short): 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: (1) 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). (2) 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 adaptive random walk strategy (ARW for short): $F(\mathcal{P} | \mathcal{P} \in \mathcal{A}_{uv}) = p_{ARW}(\mathcal{P}, v | u, \theta)$. Thus, the sampled items with ARW meets 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, 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 the subsection 3.3.

Adaption. We remark the adaption of our collaborative sampler model in following two aspects: (1) The sampling distribution of CoSam is defined with a parameterized function instead of pre-defined values. Thus, the sampling distribution will adaptively evolve with training process going on. (2) 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 [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:
 $G_u \sim \text{Multinomial}(N_u, p_s(i|u; \theta))$.
- (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 can not be differentiated by S. But the integrated framework differs from the adversarial framework in that the sampler in our framework will make 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 follow:

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

A good property of our framework can be observed: even although the recommender model is trained with an 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 with multiplied by a relatively large value; On the contrary, the prediction on these “easy” instances, which is highly negative, will be weighted with small values. Equation (9) guarantees 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:

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

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 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, otherwise they can not be selected by the 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 ; the rest 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:

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

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:

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

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

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

where we perform a sampling approximation in which the 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 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 sampler model. Due to the discrete nature of candidate set G_u , we adopt the policy gradient method from reinforced learning to optimize the model S. That is, 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 :

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

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

3.3 Collaborative sampler with integrated framework

We further integrate aforementioned collaborative sampler into our integrated framework. When collaborative sampler meets 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:

$$\begin{aligned} p_s(i|u; \theta) &= \rho_{ui} = \sum_{k \in U} \mathbf{H}_{u,k} \frac{1}{m} + \mathbf{H}_{u,n+i} \\ &= \frac{1 - c_1}{1 - c_1 c_2} \frac{1}{m} + \sum_{\mathcal{P} \in \mathcal{A}_{ui}} F(\mathcal{P}) \\ &\equiv p_0 + p_{ARW}(\mathcal{P}, i|u; \theta) \end{aligned} \quad (15)$$

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

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

where $\mathbf{I}[\cdot]$ 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:

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

The learning algorithm of CoSam is presented in Algorithm 1.

3.4 Discussion

3.4.1 Gradient analysis.

Sampling informatively. How to 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 better description. The first term in eq.(14) will give a force to push up these positive balls. Due to the correlations between the instances, which can be analogies

Algorithm 1 Learning a recommendation model with CoSam

-
- 1: Initialize parameters φ, θ randomly;
 - 2: **while** not converge **do**
 - 3: **for** each user u **do**
 - 4: Sample a set of items G_u and paths \mathcal{S}_u with adaptive random walk strategy.
 - 5: **end for**
 - 6: Calculate gradients $\nabla_{\varphi} L$ w.r.t parameters φ of R based on G_u (eq. (13)).
 - 7: Update φ based on $\nabla_{\varphi} L$.
 - 8: Calculate policy gradients $\nabla_{\theta} L$ w.r.t parameters θ of S based on \mathcal{S}_u and G_u (equation (17)).
 - 9: Update θ based on $\nabla_{\theta} L$.
 - 10: **end while**
-

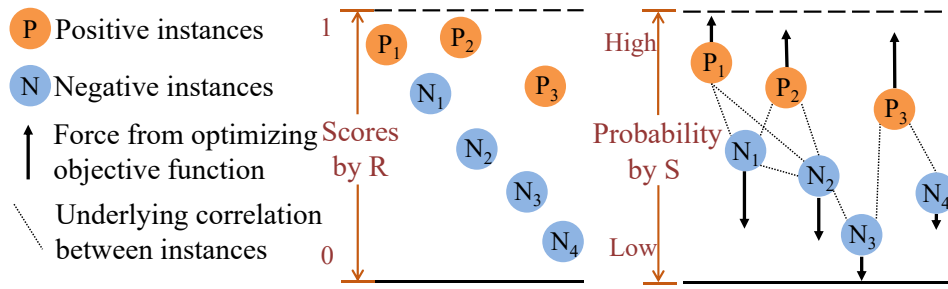


Fig. 4. Illustration of how CoSam achieves sampling informatively and collaboratively.

with elastic links lines between the balls, the negative instances which 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. On the other hand, from the second term in eq.(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 the model R, the reward will be more negative as illustrated in Figure 4. It can be explained by the collaboration effect between S and R. Note that the negative items shall be differentiated either by the model S or by the model R. If R fails to differentiate a negative item, it naturally turn to S for help and gives a signal to decrease the probability of selecting this item into 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 of the item decreases with the effect of the reward, it suggests 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: (1) 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. (2) 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 $O(n\bar{N}_u)$. (3) When updating the 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 practise, as recent works [4, 44], we usually let \bar{N}_u be several times (e.g. five) as large as the number of the positive items of the user u and thus have $n\bar{N}_u = 5|X^+|$, where $|X^+|$ denotes

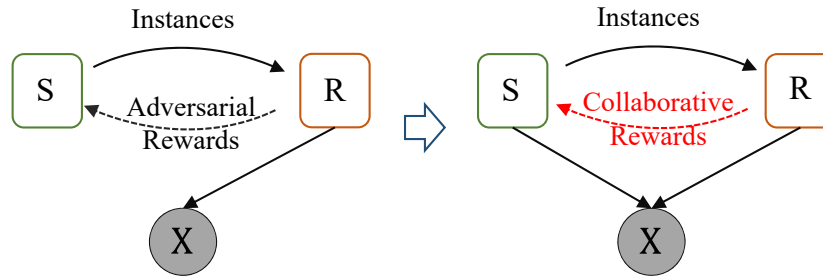


Fig. 5. Illustration of the previous adversarial framework (left) and our integrated framework (right).

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 large dataset.

3.4.3 Linking to existing methods.

Adversarial framework. We remark two key differences of our integrated framework from adversarial framework: (1) As illustrated in Figure 5, in our integrated framework, S and R collaboratively contribute to the prediction, while the prediction in adversarial framework just depends on R. (2) Both frameworks are trained in a reinforced manner. However, S in our framework is trained for collaboration with R; while S in adversarial framework is trained for competition with R.

Two-stage recommendation strategy. Another similar work is two-stage recommendation strategy [1, 6, 27, 50]. It contains: a recalling model to retrieve "positive-like" candidate items from 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 two-stage recommendation strategy are not mechanism for training a recommendation model. Instead, two-stage recommendation strategy is a operation to accelerate retrieval from a pre-trained recommendation model by reducing the scale of the ranked items. Further, from the technical view, the intermediate item set in our framework is a "soft" probabilistic variable generated from the sampled model; while the intermediate item set in the two-stage recommendation strategy is a "hard" constant selected by the retrieval model.

4 RELATED WORK

Recommendation from implicit feedback data. Recent years have seen a surge of research on recommendation techniques from implicit feedback [21, 28]. To handle the massive volume of the unobserved negative data, two strategies have been proposed in previous works. The first is employing memorization strategies to accelerate learning on the whole data. For example, ALS, eALS [2, 16, 18] have been proposed to memorize some specific intermediate variables to avoid the massive repeated computations. However, this kind of methods are just suitable for the recommendation model with K-separable property [2] and L2 loss function. The other type is sampler-based methods, where the recommendation model are trained on the reduced sampling instances with stochastic gradient descent (SGD) optimizer. This kind of methods have been widely applied in many recommendation models, including classic matrix factorization [18, 20], pair-wised models (e.g. Bpr [32], NCF [33]) and sophisticated neural network-based methods (e.g. CDL [35], DNN [41], NCF [15], XDeepFM [25], ConvNCF [14], NGCF [37], lightGCN [13], Mult-VAE [26]).

Sampler for recommendation. The most popular sampling strategy is to sample negative data with equal probability (uniformly). Although efficiently, the optimizer usually sample uninformative training instances [31, 43, 45], which makes limited contribution on updating. The model can not 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 literatures. The first is heuristic-based strategies, which pre-define a sampling rule to over-sample informative instances [17, 31, 43, 46, 48]. For example, [17, 43] propose to sample instances based on global item popularity; [31, 45, 47] propose to draw instances based on item ranking position. However, these methods requires rich human experience and also lack flexibility. Another type is model-based sampler [8, 30, 36], which defines a sampler model and the sampling probability will adaptively evolve by optimizing adversarial objective. However, as analyzed in section 2, this kind of methods will suffer from inefficiency and biases. Also, there are some works leverage auxiliary information to sample informative training instances. For example, [4, 38, 48] claim to over-sample the items that have been consumed by social friends; [7–9] leverage exposure data to find reliable negative instance. [39] leverages knowledge graph to enhance negative sampling. However, these auxiliary information may not be available in many recommendation systems.

Random walk in recommendation. Random walk strategy has been widely applied in recommendation. [19] performs random walk along the social network to search relevant users who have similar preference with the target user for better rating prediction; [5] exploits random walk to obtain diverse recommendation; [34] further extends [5] in heterogenous information network to generate valuable meta-paths; [5] performs random walk to complete implicit feedback matrix for mitigating data sparsity problem; Similarly, [45] employs 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. Besides, these random walks are not designed for sampling informative negative instances.

To our knowledge, only one recent work SamWalker [4] 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 while it in CoSam is on the user-item interaction graph; (2) the transfer probability in SamWalker is learned from users' exposure while it in CoSam 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 does different components in CoSam (i.e. collaborative sampling model and integrated framework) affect CoSam's performance?

5.1 Experimental settings

Datasets and parameters. Four well-known recommendation datasets LastFM¹, Movielens-1M², Ciao³, Epinions⁴, are used in our experiments. These datasets contain users' feedback on items. The dataset statistics are presented in Table 1. We preprocess the datasets so that all items have at least three interactions and "binarize" user's feedback into implicit feedback as [4, 40]. That is, as long as there exists some user-item interactions (ratings or clicks), the corresponding implicit feedback is assigned a value of 1. Grid search and 5-fold cross validation are used to find the best hyper-parameters and test the results. In our CoSam, as recent works [4, 44], we set $N_u = 5|\mathcal{X}_u|$ and test c_1, c_2 of [0.4, 0.6, 0.8, 1]. We also optimize the model with mini-batch Adam [22] and

¹<https://grouplens.org/datasets/hetrec-2011/>

²<https://grouplens.org/datasets/movielens/>

³<http://www.cse.msu.edu/~tangjili/trust>

⁴<http://www.trustlet.org/epinions>

Table 1. 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 feedbacks	52,627	1,000,177	138,723	333,658
Density of positive feedbacks	0.62%	4.50%	0.14%	0.05%

Table 2. The performance metrics of the compared samplers. 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.

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

test the learning rate of [0.001,0.01,0.1]. The setting of compared methods are referring to related works or tested in our experiments. All experiments are conducted on a server with 2 Intel E5-2620 CPUs, 2 NVIDIA Titan-X GPUs and 256G RAM. We will share our source code online when the paper get 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 i , we define $Rec(i)$ as the set of recommended items in top-K and $Con(i)$ as the set of consumed items in test data for user i . Then we have:

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

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

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

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

$$NDCG = \frac{1}{|U|} \sum_{i \in U} \frac{DCG_i}{IDCG_i} \quad (20)$$

where DCG_i is defined as follow and $IDCG_i$ is the ideal value of DCG_i coming from the best ranking.

$$DCG_i = \sum_{j \in Con(i)} \frac{1}{\log_2(rank_{ij} + 1)} \quad (21)$$

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

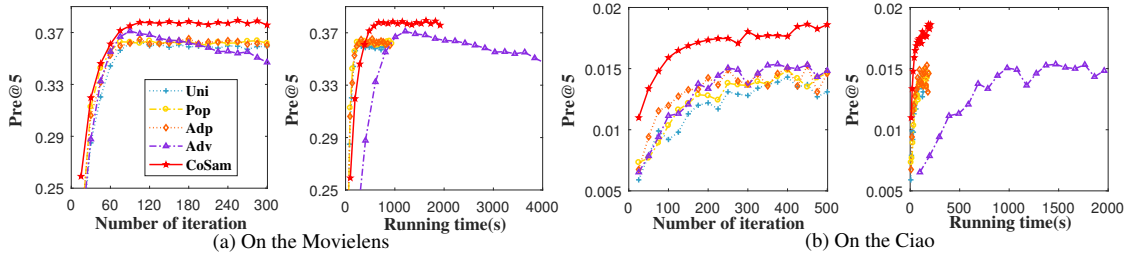


Fig. 6. Pre@5 for different sampling strategy versus the number of iterations and running time.

5.2 Comparing with existing samplers (RQ1)

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

- Uniformly (Uni): the widely adopted sampling strategy which sample items with equal probability.
- Item-Popularity (Pop) [44]: which samples items based on item popularity: $p(i|u) \propto \xi_i^\alpha$, where ξ_i denotes the popularity of item i and the parameter α scales its affect.
- Adaptive oversampling (Adp) [31]: 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 adversarial sampler-recommender framework. Since the sampler model of [8] [4] requires other side information, we follow [36] [30] and model the sampler with matrix factorisation.

Also, for fairly comparison, we integrate these samplers into common matrix factorization recommendation model. Their performance are reported in Table 2.

Performance comparison. Table 2 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 2 also show the relative improvements achieved by CoSam over the best baselines. We have following observations: (1) The informative sampler will boost the recommendation performance. It can be seen from the better performance of Adv, Ada, CoSam over Uni. (2) 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 flexible and usually fail to capture the varying informativeness of instances. (3) CoSam achieves much better performance than compared methods. Especially in the large dataset Epinions, the improvements is 61.7%, 62.7%, 3.45% in terms of Pre@5, Rec@5, NDCG respectively. It can be attributed to collaborative sampler model and integrated sampler-recommender framework. We will conduct ablation studies in the subsection 5.4 to analyze the effect of these components.

Running time. Table 2 also presents the training time cost of the compared samplers and figure 6 depicts Pre@5 (Y-axis) vs. the number of iteration or running time (X-axis) of these methods on the two typically 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 while Cosam just requires 7 mins. 20 times acceleration will be achieved by our CoSam. The speed up can be attributed to our collaborative sampler, which avoids time-consuming traversal of all items. This way, CoSam has similar time complexity as these efficient heuristical methods, which aim at fast learning but sacrifice certain flexibility. Their actual running time are also in the same magnitude.

Variance and training loss. To show the stability and informativeness of samplers, we further conduct a specific experiments on empirically comparing the variance of the estimated gradient $\nabla_L \varphi$ and average training

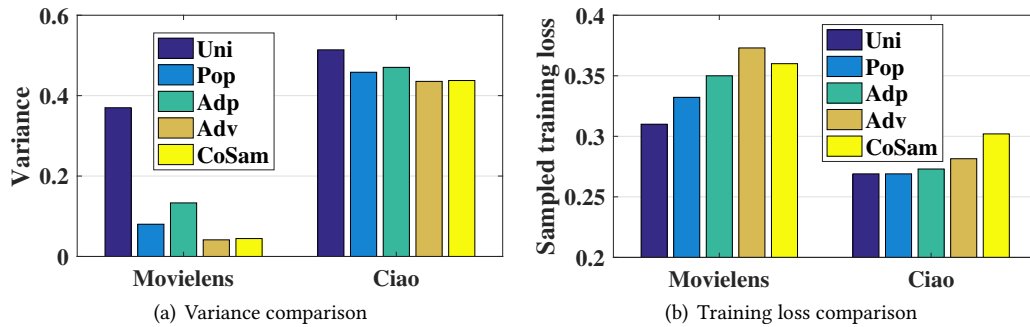


Fig. 7. Variance of gradients and sampled training loss using different sampling strategies.

loss on sampled instances. To do so, we train the models for 50,100,150 epochs and then we generate mini-batch with various sampling strategy to estimate normalized gradients and training loss. We repeat this process for 1000 times, and calculate the variance of the gradients and average training loss for different sampling strategies. The final results are averaged over various parameters and presented in Figure 7. The sampler value of variance suggests the model will be more stable, while the larger values of the sampled training loss indicates the sampled instances will be more informative. Figure 7 shows that our CoSam outperforms these heuristic samplers and achieves similar performance with Adv. We remark Adv will sacrifice training efficiency. Also, Adv will increase the risk of over-fitting as illustrated in Figure 6.

5.3 Effect of CoSam (RQ2)

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

- MF [18, 20, 29]: The classic matrix factorization model for implicit feedback data with bernoulli likelihood and uniformly sampler.
- BPR [32]: The classic pair-wise method for recommendation, coupled with matrix factorization.
- NCF: [15]: The advanced recommendation method modeling user-item interactions with neural network. We refer to [15] and train the model with uniformly sampler.
- IRGAN[36]: The method that employs 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 [37]: The state-of-the-art graph-based recommendation method, which encodes users and items based on neural graph network [12] 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 boost 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 method can beat state-of-the-art recommendation methods, even if we just employ a relatively simple recommendation model (i.e. MF). This result validate the importance of the sampling method, which makes nonnegligible contribution on recommendation performance. In fact, the recommendation model has been well exploited by substantial recent works while the sampler usually become the effect bottleneck of the recommendation system.

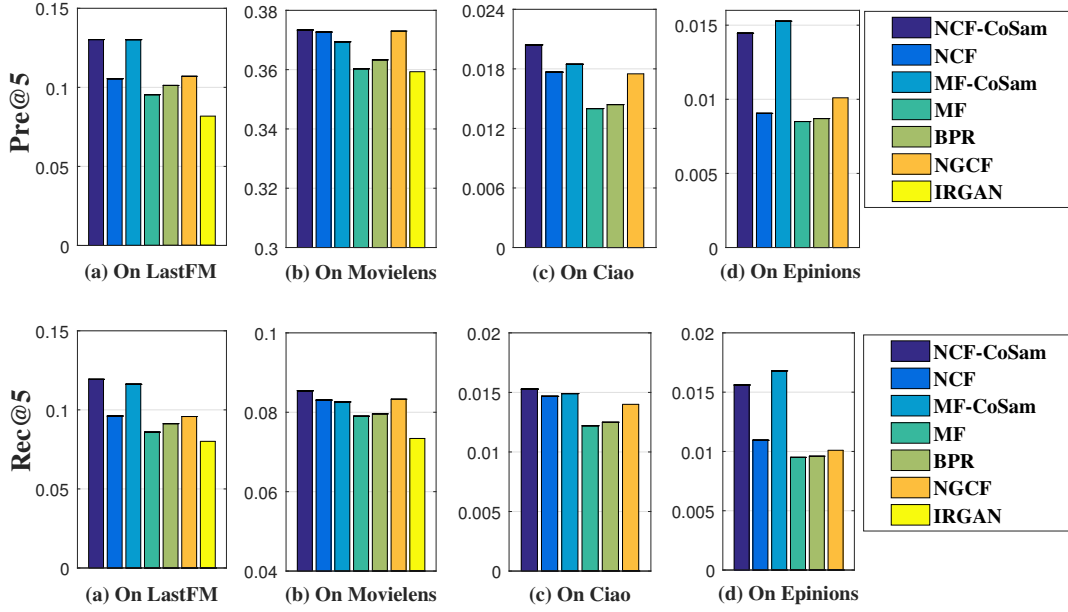


Fig. 8. Performance of different recommendation methods.

Table 3. The performance and running time of CoSam and its variants.

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

Table 4. 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	\	√

5.4 Ablation study (RQ3)

We further design a detailed ablation study to answer question RQ3. That is, we remove different components at a time and compare CoSam with its three special cases: Sam-MF-Int, Sam-Co-Adv, Sam-MF-Adv. Here we remark Sam-X-Y as the sampler with “X” sampler model and “Y” sampler-recommender framework. Here “MF” denotes matrix factorization model and “Co” denotes collaborative model; “Adv” denotes Adversarial framework and “Int”

denotes integrated framework. Note that Cosam can be remarked as Sam-Co-Int and Adv can be remarked as Sam-MF-Adv. We integrate these methods into benchmark MF recommendation model to test their performance. The recommendation performance and training running time are presented in Table 3.

We observe that the method with integrated sampler-recommender framework (Sam-Co-Int, Sam-MF-Int) outperforms their variants with adversarial framework (Sam-Co-Adv, Sam-MF-Adv). This is due to the ability of the sampler model has been better exploited and the sampling biases have been refined in the collaborative framework. For the sampler models comparison, generally, the collaborative sampler model outperforms MF-based sampler in the collaborative framework, while it performs worse in the adversarial framework. This interesting phenomenon can be explained as follow: collaborative sampler, which encodes rich interaction information into sampling, will produce more informative instances. Naturally, it will achieves better performance under integrated framework. However, when it is applied into 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 efficient than MF-based sampler model. Thus, collaborative sampler will be more suitable for the modern recommendation system with large item space.

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

- Sam-MF-Int-S, Sam-MF-Int-R: The special cases of Sam-MF-Int, which trains S and R based on integrated framework but make 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 adversarial framework but unifies S and R for recommendation as integrated framework.

Their characteristics are presented in Table 4. Here we choose Sam-MF-Int instead of Sam-Co-Int (CoSam) for experiments to isolate the effect of 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 make 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” will be deteriorated; Similarly, S is not trained to capture users’ preference individually and thus Sam-MF-Int-S performs inferior to the many recommendation method. Only when R meets S in recommendation, they will compensate each other and achieve impressive performance. However, we observe that unifying R and S into recommendation with adversarial framework is ineffective. Sam-MF-adv-U performs worse instead of better than Sam-MF-adv. It can be attributed to the different rewards (as discussed in subsection 3.4). In adversarial framework, S is trained for competition instead of collaboration with R. S and R are not ready for collaborative prediction.

6 CONCLUSION

In this paper, we present a novel efficient and effective sampling method CoSam. It consists of: (1) a collaborative sampler model which leverages collaborative signals into sampling and satisfies the desirable property of normalization, adaption, interaction information awareness, and computational efficiency; (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 sophisticated graph neural network in sampler process, to capture more complex affinity between users and items along the interaction graph. Also, effectiveness and efficiency are dual objectives for sampling strategies. It will be interesting and challenging to design more

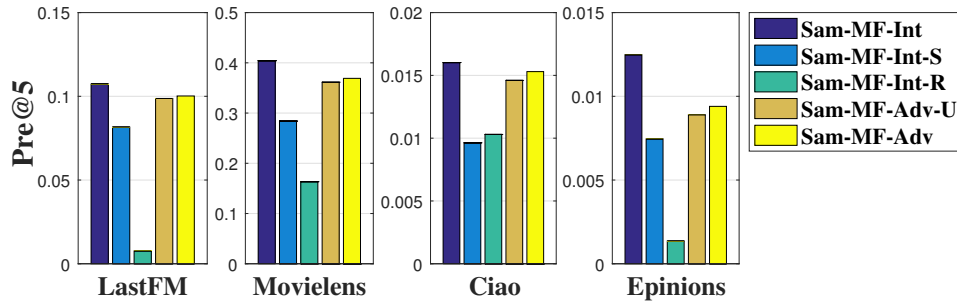


Fig. 9. Performance of Sam-MF-Int and its variants.

advanced sampler meeting both conditions. Another interesting direction is to explore dynamic sampling strategy in online recommender system, where user preferences will drift with time[24].

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China (Grant No: 2018AAA0101505, 2019YFB1600700) and National Natural Science Foundation of China (Grant No: U1866602).

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, 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*. IW3C2, 1341–1350.
- [3] Jeffery P Bray. 2008. Consumer behaviour theory: approaches and models. (2008).
- [4] Jiawei Chen, Can Wang, Sheng Zhou, Qihao Shi, Yan Feng, and Chun Chen. 2019. SamWalker: Social Recommendation with Informative Sampling Strategy. In *The World Wide Web Conference*. ACM, 228–239.
- [5] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. 2015. Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks. *Conference on Recommender Systems* (2015).
- [6] 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, 191–198.
- [7] 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 of the The Web Conference 2018 on The Web Conference 2018*. IW3C2, 13–14.
- [8] Jingtao Ding, Yuhuan 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*. AAAI Press, 2230–2236.
- [9] 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* (2019).
- [10] Hongchang Gao and Heng Huang. 2018. Self-paced network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1406–1415.
- [11] 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.
- [12] Will Hamilton, Zhitaoying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
- [13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2002.02126* (2020).
- [14] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912* (2018).
- [15] 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, 173–182.

- [16] 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, 549–558.
- [17] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Stochastic inference for scalable probabilistic modeling of binary matrices. In *International Conference on Machine Learning*. 379–387.
- [18] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.
- [19] 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, 397–406.
- [20] Christopher C Johnson. 2014. Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems* 27 (2014).
- [21] Diane Kelly. 2005. Implicit feedback: Using behavior to infer relevance. In *New directions in cognitive information retrieval*. Springer, 169–186.
- [22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [23] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [24] 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.
- [25] 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 & Data Mining*. ACM, 1754–1763.
- [26] 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*. International World Wide Web Conferences Steering Committee, 689–698.
- [27] 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 preprint arXiv:1907.06323* (2019).
- [28] Douglas W Oard, Jinmook Kim, et al. 1998. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, Vol. 83. WoUongong.
- [29] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. *ICDM (2008)*, 502–511. <https://doi.org/10.1109/ICDM.2008.16>
- [30] Dae Hoon Park and Yi Chang. 2019. Adversarial Sampling and Training for Semi-Supervised Information Retrieval. In *The World Wide Web Conference*. ACM, 1443–1453.
- [31] 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, 273–282.
- [32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.
- [33] 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, 1353–1362.
- [34] Fatemeh Vahedian, D. Robin Burke, and Bamshad Mobasher. 2017. Weighted Random Walk Sampling for Multi-Relational Recommendation. *UMAP* (2017).
- [35] 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, 1235–1244.
- [36] 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.
- [37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. *arXiv preprint arXiv:1905.08108* (2019).
- [38] 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, 5–14.
- [39] 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 Web Conference 2020*. 99–109.
- [40] 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 on Conference on Information and Knowledge Management*. ACM, 337–346.
- [41] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*. 3203–3209.

- [42] 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 & Data Mining*. 974–983.
- [43] 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.
- [44] Hsiang-Fu Yu and Chih-Jen Lin. 2017. Selection of Negative Samples for One-class Matrix Factorization. *SIAM International Conference on Data Mining (SDM) (2017)*, 1–27.
- [45] Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang. 2018. Walkranker: A unified pairwise ranking model with multiple relations for item recommendation. AAAI.
- [46] 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, 45–54.
- [47] 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.
- [48] 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 Conference on Information and Knowledge Management*. ACM, 261–270.
- [49] 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.
- [50] 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 & Data Mining*. ACM, 1079–1088.