PURE: Positive-Unlabeled Recommendation with Generative Adversarial Network

Yao Zhou[†], Jianpeng Xu[§], Jun Wu[†], Zeinab Taghavi[§], Evren Korpeoglu[§]
Kannan Achan[§], Jingrui He[†]
[†]University of Illinois at Urbana Champaign, {yaozhou3, junwu3, jingrui}@illinois.edu;
[§]Walmart Labs, {jianpeng.xu, ZTaghavi, EKorpeoglu, kannan.achan}@walmart.com

ABSTRACT

Recommender systems are powerful tools for information filtering with the ever-growing amount of online data. Despite its success and wide adoption in various web applications and personalized products, many existing recommender systems still suffer from multiple drawbacks such as large amount of unobserved feedback, poor model convergence, etc. These drawbacks of existing work are mainly due to the following two reasons: first, the widely used negative sampling strategy, which treats the unlabeled entries as negative samples, is invalid in real-world settings; second, all training samples are retrieved from the discrete observations, and the underlying true distribution of the users and items is not learned.

In this paper, we address these issues by developing a novel framework named PURE, which trains an unbiased positive-unlabeled discriminator to distinguish the true relevant user-item pairs against the ones that are non-relevant, and a generator that learns the underlying user-item continuous distribution. For a comprehensive comparison, we considered 14 popular baselines from 5 different categories of recommendation approaches. Extensive experiments on two public real-world data sets demonstrate that PURE achieves the best performance in terms of 8 ranking based evaluation metrics.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender systems, Positive-unlabeled learning

ACM Reference Format:
Yao Zhou[†], Jianpeng Xu[§], Jun Wu[†], Zeinab Taghavi[§], Evren Korpeoglu[§], Kannan Achan[§], Jingrui He[†]. 2021. PURE: Positive-Unlabeled Recommendation with Generative Adversarial Network. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.*, 11 pages. https://doi.org/10.1145/3447548.3467234

1 INTRODUCTION

Recommender systems have been prevalent in recent decades across multiple domains in e-Commerce [41], content streaming (YouTube) [6], and business service industries (Yelp) [37], due to their success in filtering or retrieving relevant information from user profiles and behaviors. Traditional collaborative filtering methods [11, 25] and matrix factorization methods [23, 28, 35] are the most popular and effective methods of recommender systems for many years. Lately, various embedding based methods such as deep factorization machine [13] and neural collaborative filtering [15] have been proposed and achieved a lot of successes. This leads to a wide and in-depth study of the deep learning based recommender systems [44]. Most of the existing methods take the following two assumptions for granted, especially for *implicit* recommender systems: (1) The unobserved interactions between users and items (i.e., unlabeled user-item tuples) are often labeled as negative samples; (2) The observed users, items, and their interactions are representing the true relevance distribution. However, these assumptions are usually not valid for real-world recommender systems.

In the first assumption, it assumes that an item i is more relevant to a user u than item j if i has interactions with u while j does not. The assumption is not necessarily true in that, the missing of interactions between item j and user u could be because of the lack of the exposure between item j and user u, rather than the uninterestingness of u on j. In other words, the unlabeled user-item tuple can be either a positive or negative sample. Hence, simply using the unlabeled tuples as negative samples in the training process can inevitably degrade the model performance. In this paper, instead of taking the unlabeled tuples as negative samples, we formulate the recommender system into a Positive-Unlabeled (PU) learning [2] framework, which is a machine learning approach where the learner observes only positive data and unlabeled data.Existing works of PU learning mainly focus on designing the PU learning adapted objectives [30]. It has been theoretically analyzed in [21] that for unbiased PU learning, the empirical risks on training data can be negative if the training model is very flexible, which will result in serious overfitting. Hence, even though flexible models such as deep neural networks have been widely explored in recommender systems, limited work has been done under the PU learning setting.

Secondly, in traditional recommender systems, the training samples are usually composed of the positive (labeled) samples and a sampled set of negative samples from the unlabeled data. This negative sampling process can be problematic in that, as we mentioned in PU learning, the samples from the unlabeled data may not necessarily be the real negative ones, and this will distort the learned data distribution in the modeling process. Generative models such as generative adversarial networks (GAN) [12] tried to alleviate the issue of negative sampling by learning the underlying data distribution from an implicit generative model instead of

KDD '21, August 14–18, 2021, Virtual Event, Singapore.

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467234

imposing any assumption on the existing data. In the framework of GAN, a discriminator is introduced to distinguish the generated samples of the generator from the real samples, while the generator is optimized in such a way that its generated samples are hardly separable by the discriminator. Specifically, IRGAN [40] was proposed to apply GAN on learning-to-rank applications where it employs policy gradient based reinforcement learning to perform discrete sampling of documents (items) for each query (user), in order to select relevant items from a given pool. However, we argue that this discrete sampling strategy may limit the expressiveness of the generator due to the sparsity of data in recommendation, and the model will not learn the underlying true distribution of the users and items. Besides, IRGAN only performed sampling on items, and used all users in the loss function, which makes IRGAN lack the capability to learn the distribution of the users.

In order to address the aforementioned limitations of existing works, we propose a novel approach called **P**ositive-**U**nlabeled **RE**commendation with generative adversarial network (PURE). First of all, based on the analysis of [21], PURE adopts the *positive unlabeled risk minimizer* to train an unbiased positive-unlabeled discriminator. In particular, we theoretically prove that the estimation error bound of PU Learning is tighter than that of positive-negative (PN) learning when the number of unlabeled samples is lower bounded, which can be easily satisfied due to the extreme sparsity of the real-world data. In addition, in order to learn the true distribution of users and items, continuous sampling on both users and items in the embedding space is employed in the generator. Specifically, a fake item (embedding) for a user is generated with a random noise input. A fake user (embedding) can also be generated in a similar way. Furthermore, we theoretically prove that the optimal generator is able to generate high-quality embeddings from a learned user-item distribution that is very similar to the true user-item relevance distribution. The main contributions of this paper are summarized below:

- We propose a novel approach for recommender systems called PURE under the GAN framework, which trains an unbiased positive-unlabeled discriminator using PU learning.
- The generator of PURE performs continuous sampling on both users and items in the embedding space in order to learn the true relevance distribution of the users and items.
- We theoretically prove that an unlabeled sampling bound of PURE exists and can be satisfied easily in real-world recommender systems. The optimalities upon convergence are also provided for both the discriminator and the generator.

The rest of the paper is organized as follows. Section 2 is the preliminary. Section 3 describes the proposed framework PURE, and Section 4 presents the analyses of PURE from various perspectives. The experimental results are illustrated in Section 5. In Section 6, we briefly introduce the related work on recommender systems and PU learning. In the end, we conclude the paper in Section 7.

2 PRELIMINARY

In this section, we first present the notation as well as the problem definition for recommendation. Then, the preliminary work of generalized matrix factorization (GMF) and generative adversarial network (GAN) are briefly reviewed.

2.1 Problem Definition

In this paper, we study the implicit recommendation problem and let $\mathcal{U}$ and $\mathcal{I}$ denote the sets of users and items. Given a user u, a list of relevant items can be rated or viewed by u. From the perspective of matrix representation, we define the user-item interaction matrix as $\mathcal{R} \in \{1, 0\}^{M \times N}$, where M and N denote the number of users and items, respectively. The entry $\mathcal{R}_{ui} = 1$ if there is an observed interaction between user u and item i. We further assume Ω to be the index set of these observed entries, namely, $(u, i) \in \Omega$ if $\mathcal{R}_{ui} = 1$. It should be noticed that $\mathcal{R}_{ui} = 0$ does not necessarily mean that user u dislikes item i. The unobserved entries could be missing data with either positive labels (i.e., user and item are truly relevant) or negative labels (i.e., user and item are non-relevant). In real applications, each user can only rate and view a very limited number of items. Therefore, without loss of generality, we assume the truly relevant user-item tuples are very sparse in nature. Then, the recommendation problem is usually formulated as follows:

Definition 1 (**Recommendation Problem**).
Given: *A set of users $\mathcal{U} = \{u_1, u_2, ..., u_M\}$, a set of items $\mathcal{I} = \{i_1, i_2, ..., i_N\}$, the observed user-item interaction matrix $\mathcal{R}$.*
Output: *The estimated interaction scores of the unobserved items for each user u in $\mathcal{U}$.*

2.2 Generalized Matrix Factorization

Matrix Factorization (MF) is one of the most successful recommendation approaches that realize the latent factor models by decomposing the user-item matrix $\mathcal{R}$ into the product of two lower dimensional matrices. The MF model usually maps both users and items to a joint latent factor space with the dimensionality of d. Accordingly, each user u is associated with a latent vector $\boldsymbol{e}_u \in \mathbb{R}^d$, and each item i is associated with a latent vector $\boldsymbol{e}_i \in \mathbb{R}^d$. To learn these latent factor vectors, the objective is usually designed to minimize the squared error on the observed user-item tuples:

$$\min_{\{\boldsymbol{e}_u, \boldsymbol{e}_i\}} \sum_{(u,i) \in \Omega} (\mathcal{R}_{ui} - \boldsymbol{e}_u^\top \boldsymbol{e}_i)^2 \tag{1}$$

Despite its success in various applications, MF assumes user and item latent features are equally important on each dimension, and combines them with equal weights. However, [15] has pointed out that MF can incur a large ranking error due to its naive assumption. Therefore, they propose to use a GMF model to increase the expressiveness of MF:

$$\min_{\{\boldsymbol{e}_u, \boldsymbol{e}_i\}} \sum_{(u,i) \in \Omega \cup \Omega^-} \left(\mathcal{R}_{ui} - \{\boldsymbol{e}_u \odot \boldsymbol{e}_i\}^\top \boldsymbol{r}_D\right)^2 \tag{2}$$

where $\odot$ is the element-wise product and Ω^- denotes the set of negative samples, which are sampled from the unobserved user-item interactions. $\boldsymbol{r}_D$ is a learnable vector which builds the relation mapping between user latent vector $\boldsymbol{e}_u$ and item latent vector $\boldsymbol{e}_i$.

2.3 Generative Adversarial Network

GAN was initially introduced in [12] and it consists of two models, i.e., discriminator D and generator G, that play a minimax game. The discriminator D aims to distinguish the real-world data and the fake data from the generator G. Meanwhile, the generator G

aims to generate fake data to confuse the discriminator D as much as possible. The objective of GAN is usually formatted as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{p_{data}(x)}\Big[\log D(x)\Big] + \mathbb{E}_{p_g(x)}\Big[\log(1-D(x))\Big] \tag{3}$$

where $p_{data}(x)$ and $p_g(x)$ represent the distributions of real-world data and generator G's fake output data. The objective of GAN is equivalent to minimizing the Jensen-Shannon Divergence between $p_{data}(x)$ and $p_g(x)$. Therefore, upon convergence, we expect G to generate high-quality fake data that are visually similar to the real data. The problem in Eq. (3) is the conceptual formulation of GAN that favors the theoretical analysis, however, in implementation, we still need to include the objective function for loss calculation and gradient back-propagation. Then, the objective becomes:

$$\min_G \max_D V(D,G) = \mathbb{E}_{p_{data}(x)}\Big[f_D(D(x))\Big] + \mathbb{E}_{p_g(x)}\Big[f_G(D(x))\Big] \tag{4}$$

where f_D and f_G are the loss functions for discriminator D and generator G, respectively.

3 PROPOSED APPROACH

Similar to GAN, we first present the discriminator in PURE which has the ability to take various types of training samples into consideration under the PU learning setting. Then, we introduce the generator which could generate the fake user and fake item embeddings that increases model expressiveness by covering the corners of the feature space. The overview of the PURE is shown in Figure 1.

3.1 PU Classifications in Recommendation

In recommendation, we usually learn to map each user-item tuple (u,i) to a scalar value that can represent the relevance of i to u. In our framework, we design the discriminator $D(u,i)$ to be able to maps (u,i) to the value of $Y \in \{0,1\}$. The goal of the discriminative model is to distinguish between the truly relevant items and non-relevant items for the given user. Intuitively, the discriminator $D(u,i)$ is simply a binary classifier that outputs a probability relevance score. This output score should be 1 when the item i is truly relevant to the user u, and should be 0 when u and i are non-relevant. Formally, we quantify the output score of the discriminator as:

$$D(u,i) = \frac{1}{1+\exp\Big(-\phi(u,i)\Big)} \tag{5}$$

where we let $\phi(u,i) : \mathbb{N}\times\mathbb{N} \to \mathbb{R}$ be the decision function of the discriminator $D(u,i)$ and $\mathbb{N}$ is the set of natural numbers for user and item indices. The specific instantiation of decision function $\phi(u,i)$ can be versatile (e.g., matrix factorization [23], factorization machine [33], neural networks [15], etc.).

We let $p_{data}(u,i)$ be the underlying joint distribution of users and items, and $\pi_p = p(Y=1)$ be the positive class prior. Then, this joint distribution can be rewritten as follows based on the law of the total probability:

$$p_{data}(u,i) = \pi_p p_p(u,i) + (1-\pi_p)p_n(u,i) \tag{6}$$

Here, the positive user-item tuples are assumed to be drawn from the positive marginal distribution $p_p(u,i) = p_{data}(u,i|Y=1)$, and the negative tuples are drawn from the negative marginal distribution $p_n(x) = p_{data}(u,i|Y=0)$.

To train the recommendation model, we let $L(\hat{y},y)$ be the loss function, where y is the ground truth and $\hat{y}$ is the prediction. Then, the expected learning risk of the discriminator is $R(D) = \mathbb{E}_{p_{data}(u,i)}\Big[L\Big(D(u,i),Y\Big)\Big]$. Thereby, a positive-negative (PN) risk minimizer for D can be learned as:

$$\min_D R(D) = \pi_p R_p^+(D) + (1-\pi_p)R_n^-(D) \tag{7}$$

where $R_p^+(D) = \mathbb{E}_{p_p(u,i)}\Big[L\Big(D(u,i),1\Big)\Big]$ is the risk of the relevant samples w.r.t. the positive labels ($Y=1$) and $R_n^-(D) = \mathbb{E}_{p_n(u,i)}\Big[L\Big(D(u,i),0\Big)\Big]$ is the risk of the non-relevant samples w.r.t. the negative labels ($Y=0$). In practice, $R_p^+(D)$ can be approximated empirically using the observed relevant user-item tuples, but $R_n^-(D)$ is usually unknown. To estimate the learning risk, many existing work simply assume the set of the unobserved user-item tuples from the unlabeled distribution $p_u(u,i)$ are non-relevant, and perform negative sampling by assigning these tuples with negative labels.

Nevertheless, this assumption can hardly be satisfied in real scenarios since such "negatively" sampled data will inevitably include a certain number of positive samples. Naively assigning them with negative labels, the training process of the recommender system is usually unstable and often has poor convergence [34]. To this end, PU learning [21, 30] can be used to tackle this problem with theoretical guarantees by treating the unobserved user-item tuples directly as unlabeled samples. Following [21], we also express the unlabeled marginal distribution as $(1-\pi_p)p_n(u,i) = p_u(u,i) - \pi_p p_p(u,i)$. Then, $R_n^-(D)$ has the following equality:

$$(1-\pi_p)R_n^-(D) = R_u^-(D) - \pi_p R_p^-(D) \tag{8}$$

where $R_u^-(D) = \mathbb{E}_{p_u(u,i)}\Big[L\Big(D(u,i),0\Big)\Big]$ is the risk of unlabeled samples w.r.t. the negative labels, and $R_p^-(D) = \mathbb{E}_{p_p(u,i)}\Big[L\Big(D(u,i),0\Big)\Big]$ is the risk of positive samples w.r.t. the negative labels. Thus, the final risk minimization problem can be rewritten as:

$$\min_D R(D) = \pi_p R_p^+(D) - \pi_p R_p^-(D) + R_u^-(D) \tag{9}$$

By minimizing the objective of Eq. (9), the discriminator D can distinguish the relevance of user-item tuples by minimizing the learning risks of $p_p(u,i)$ and $p_u(u,i)$. Note that due to the negative property of the second term in Eq. (9), many existing work [14, 21] may replace it with $\max\{0, -\pi_p R_p^-(D) + R_u^-(D)\}$ to guarantee a non-negative risk. However, in recommendation, the positive class prior π_p is always very small which alleviates this issue, and we did not observe such a negative risk phenomenon in our experiments without adding the max operator.

3.2 Discriminative Model

With the well-defined risk minimization objective, now we demonstrate how to empirically train the discriminator using the following sets of training samples:

Positive samples from given observations. User u and item i are observed in the given data set and are truly relevant ($\mathcal{R}_{ui} = 1$).

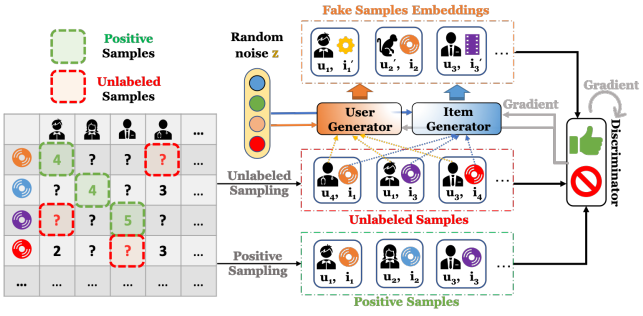

Figure 1: Overview of the proposed PURE framework

For these samples, the discriminator aims to maximize the following objective:

$$V(D)_1 = \sum_{(u,i)\in\Omega}^{n_p} \pi_p \log D(u,i) - \pi_p \log\Big(1 - D(u,i)\Big) \tag{10}$$

where $n_p = |\mathcal{R}|$ is the number of observed positive tuples. To comply with the PU learning objective in Eq. (9), the second term is the empirical risk of positive samples w.r.t. negative labels. Intuitively, we want to maximize (minimize) the D's predictions on samples with positive (negative) labels.

Unlabeled samples from unobserved interactions and the generator. Given a user u, the discriminative model is designed to assign lower scores to the items that have not be rated or viewed. We decompose this part of the objective from both the unobserved samples and generated user-item samples:

$$V(D)_2 = \sum_{(u,i)\in\Omega^-}^{n_u} \log\Big(1-D(u,i)\Big) + \Big[\log\Big(1-D(u,i')\Big) + \log\Big(1-D(u',i)\Big)\Big] \tag{11}$$

where the fake user $u' \sim G(z_u)$ and fake item $i' \sim G(z_i)$ are generated from the user and item generators respectively, and n_u is the number of unlabeled tuples from unlabeled sampling. The ratio between the unlabeled samples generated by the generator and sampled from unobserved tuples could be a hyper-parameter to tune. Here, we set their ratio to be 1 in the experiments, namely, these two sources of unlabeled samples are equally important. However, further tuning of this ratio may lead to better performance, and we leave it for future exploration.

3.3 Generative Model

The generative model aims to generate fake samples to fool the discriminator as much as possible. Therefore, given a real sample (u,i), the generator $G_i(z_i)$ is designed to generate a fake item i' that is highly likely to be relevant to u. This fake item can be virtual, and do not even exist in $\mathcal{I}$. Similarly, the generator $G_u(z_u)$ will generate a fake user u' that is likely to be relevant to i. In particular, we design the noise input for user and item generators to be a random Gaussian noise:

$$z_i, z_u \sim \mathcal{N}(\mathbf{0}, \delta \boldsymbol{I}) \tag{12}$$

where the mean of noise input would be a zero vector $\mathbf{0}$ of the same size as embedding dimension d, and $\boldsymbol{I} \in \mathbb{R}^{d\times d}$ is the identity matrix whose magnitude is controlled by δ which represents the underlying deviations of the generator's noise input. Next, we apply the multi-layer perceptron (MLP) to generate the fake item i' and user u' as follows:

$$\begin{aligned} i' \sim G_i(z_i) &= \text{ReLU}\Big(W_i^2 \cdot \text{ReLU}\Big(W_i^1 \cdot z_i + b_i^1\Big)\Big) + b_i^2\Big) \\ u' \sim G_u(z_u) &= \text{ReLU}\Big(W_u^2 \cdot \text{ReLU}\Big(W_u^1 \cdot z_u + b_u^1\Big)\Big) + b_u^2\Big) \end{aligned} \tag{13}$$

where W_i^1, W_i^2 and b_i^1, b_i^2 are the learnable weights and biases for the 1-st layer and the 2-nd layer of MLP in the item generator $G_i(z_i)$, and we have similar definitions for the user generator $G_u(z_u)$. In the experiments, we observe that a two-layer MLP would be very effective and computationally efficient. Then, putting everything together, we have the overall objective of PURE as follows:

$$\begin{aligned} \min_G \max_D V(D,G) = &\sum_{(u,i)\in\Omega}^{n_p} \pi_p \log D(u,i) - \pi_p \log\Big(1 - D(u,i)\Big) \\ + &\sum_{(u,i)\in\Omega^-}^{n_u} \log\Big(1 - D(u,i)\Big) + \Big[\log\Big(1 - D(u, G_i(z_i))\Big) + \log\Big(1 - D(G_u(z_u), i)\Big)\Big] \end{aligned} \tag{14}$$

The above objective can be optimized by performing a gradient-based optimization method. We find that Adam [20] would be empirically more stable and converge faster than other optimizers.

4 MODEL ANALYSIS

4.1 Instantiation of the Discriminator

For discriminator's decision function $\phi(u,i)$, we can define it in various ways. In our experiment, we adopt the design of GMF (see Eq. (2)) by assuming that user and item embeddings have the same dimensionality:

$$D(u,i) = \frac{1}{1 + \exp(-\{\boldsymbol{e}_u \odot \boldsymbol{e}_i\}^\top \boldsymbol{r}_D)} \tag{15}$$

In practice, we set the user embedding and item embedding to have the same dimension. Nevertheless, it is rather straightforward to extend it to a more general setting that users and items have different embedding dimensions. Then, a more generalized form for quantifying the output score of discriminator D is:

$$D(u,i) = \frac{1}{1 + \exp(-\boldsymbol{e}_u^\top M_D \boldsymbol{e}_i)} \tag{16}$$

where $\boldsymbol{e}_u \in \mathbb{R}^{d_u}$ and $\boldsymbol{e}_i \in \mathbb{R}^{d_i}$ are the latent embeddings of user u with size d_u and item i with size d_i, respectively. $M_D \in \mathbb{R}^{d_u \times d_i}$ is the learnable relation mapping matrix for user and item embeddings. Note that MF-based and GMF-based discriminators are both special cases of Eq. (16) by setting $d = d_u = d_i$ and M_D as an identity matrix or a diagonal matrix.

4.2 Sampling Strategy

In PN learning, it is a common practice to treat the observed user-item tuples as positive, and treat the rest as negative. However, due to the sparsity of the positive tuples, we frequently sample the negative tuples from a large number of unlabeled entries. One popular sampling strategy is uniform negative sampling (UNS), where the number of sampled "negative" tuples n_n is proportional to the number of positive tuples n_p. Nevertheless, UNS may lead to poor and unstable convergence [34] during training due to its ill-conditioned assumption. To stabilize and improve the model

performance, other techniques have been developed to alleviate the convergence issue, such as dynamic negative sampling (DNS) or dynamic random negative sampling (DRNS) [40]. Their intuitions are similar to the concept of one-class SVM [29], which wraps a classification boundary around the positive samples and treats the rest as negative. Both DNS and DRNS have been shown to be faster in terms of model convergence [3, 34, 40] in the PN learning setting. However, both of them need to call the learned model repeatedly which is extremely computational expensive especially for large-scale data sets. In PURE, we adopt the efficient UNS sampling strategy since unlabeled data have been explicitly modeled in our PU learning objective.

4.3 Sampling Bound

Another key question is how to determine the number of unlabeled samples n_u. In PN learning, the selection of n_n is usually empirical, where $n_n = Cn_p$ and C is the negative sampling ratio. However, in PU learning, with the utilization of estimation error bound [30], n_u can be determined by π_p and n_p using the following theorem.

Theorem 1. *[**Unlabeled Sampling**] The estimation error bound of PU learning is tighter than that of PN learning if and only if:*

$$n_u \geq \frac{\sqrt{C}\, n_p}{\left(1 - \left(\sqrt{C}+1\right)\pi_p\right)^2} \tag{17}$$

Intuitively, n_u monotonically decreases with a decreasing π_p, and a larger C in PN learning will require a larger n_u to guarantee that PU learning outperforms PN learning. In practice, π_p must be much smaller than 0.5 because positive samples are very sparse in recommendation. As a special case, we can set $C = 1$ which means the negative sampling in PN learning follows the $1:1$ balanced setting. Then, from $n_u \geq n_p/(1-2\pi_p)^2$, we easily know that when π_p is small, e.g., less than 0.1, PU learning is expected to outperform the corresponding PN learning with $n_u = 2n_p$. When π_p increases, e.g., greater than 0.4, PU learning is difficult to beat PN learning unless $n_u \geq 25n_p$. Namely, when π_p is small (which is mostly the case for recommendation problems), PU learning is a better and computationally efficient option.

4.4 Optimality of Convergence

Up to now, it is still unclear whether the final convergence of PURE would enjoy the desirable property of our initial motivation of having a good generator to produce high-quality fake sample embeddings. In this section, we provide theoretical proof to show that the objective of PURE is equivalent to minimizing the KL-divergence between the true user-item relevant distribution $p_p(u,i)$ and generated distribution $p_g(u,i)$ of the generator plus unlabeled distribution $p_u(u,i)$.

First, following the analysis in [12], we show that the optimal distribution of discriminator D would be a balance between $p_p(u,i)$, $p_n(u,i)$, and $p_g(u,i)$.

Proposition 1. *[**Optimality of the discriminator**] For a fixed generator G, the optimal discriminator D is:*

$$D^*(u,i) = \frac{\pi_p p_p(u,i)}{p_u(u,i) + p_g(u,i)}$$

Algorithm 1 PURE

Input: Generators G_u, G_i, discriminator D, user-item interaction matrix $\mathcal{R}$, user set $\mathcal{U}$, item set $\mathcal{I}$, positive class prior π_p.
Initialization: Assign G_u, G_i with random weights, assign D with random weights or pre-trained weights, $n_p = |\Omega|$, $n_u = \text{ceil}\left(\frac{n_p}{(1-2\pi_p)^2}\right)$
Repeat:
  for discriminator-steps **do:**
    Sample first n_p tuples $(u,i) \in \Omega$ with label 1
    Sample another n_p tuples $(u,i) \in \Omega$ with label 0.
    Sample n_u tuples $(u,i) \in \Omega^-$ with label 0.
    Generate n_u tuples (u,i') and (u',i) with label 0 using Eq. (13).
    Update the discriminator model D by ascending its gradients in the objectives of Eq. (10), and Eq. (11).
  end for
  for generator-steps **do:**
    Generate n_u random noise z_u, z_i using Eq. (12).
    Sample n_u tuples $(u,i) \in \Omega^-$ with label 1.
    Replace (u,i) with (u,i') and (u',i) using generator's output $G_u(z_u)$ and $G_i(z_i)$ by Eq. (13)
    Update the generator model G_u, G_i by descending their corresponding gradients in the objective of Eq. (11).
  end for
Output: The trained G_u, G_i, and D

Next, with the optimal discriminator being fixed, we can substitute $D^*(u,i)$ into the final objective of PURE in Eq. (14). Then, we can have the optimal generator as follows.

Proposition 2. *[**Optimality of the generator**] With the discriminator D fixed, the optimization of the generator is equivalent to minimizing: $-2\mathrm{H}\left(\frac{\pi_p}{2}\right) + \pi_p \cdot \mathrm{KL}\left(p_p(u,i) \middle\| \frac{p_u(u,i)+p_g(u,i)}{2}\right) + (2-\pi_p) \cdot \mathrm{KL}\left(\frac{(1-\pi_p)p_n(u,i)+p_g(u,i)}{2-\pi_p} \middle\| \frac{p_u(u,i)+p_g(u,i)}{2}\right)$ where $\mathrm{H}\left(\frac{\pi_p}{2}\right)$ is the entropy for a Bernoulli with success probability of $\frac{\pi_p}{2}$.*

Theorem 2. *[**Global optimum**] The global minimum could be achieved if and only if $p_p(u,i) = \frac{p_u(u,i)+p_g(u,i)}{2}$. At that point, the objective value of the framework $V(G,D)$ converges to $-2\mathrm{H}\left(\frac{\pi_p}{2}\right)$, and the value of $D(u,i)$ reaches $\frac{\pi_p}{2}$.*

The proofs of the above four theoretical results can be found in the Appendix. In Theorem 2, we know the proposed framework will achieve equilibrium if and only of $p_p(u,i) = \frac{p_u(u,i)+p_g(u,i)}{2}$. Intuitively, upon convergence, linearly combining the optimal generator's user-item distribution with the original unlabeled user-item distribution of the given data, will be highly similar to the true relevant user-item distribution. This justifies our motivation for training a generator to produce highly relevant embeddings that confuse the discriminator as much as possible.

4.5 Algorithm and Complexity

Based on the overall learning objective, we summarize the learning steps of PURE in Algorithm 1. Before training, the generator

and the discriminator are initialized either randomly or with pre-trained weights. Then, during the training stage, we update these two models respectively in an iterative manner. Specifically, we first fix G and update the discriminator using the observed positive tuples, the sampled unlabeled tuples[1], and the generated user/item embeddings. Next, we fix D and update the generator. The aforementioned iterative steps will continue until the model converges or the max number of iterations is reached.

Regarding the complexity analysis of the model training, we assume both user and item have equal latent embedding dimensionality d. Then, the space complexity is $O\big((M+N+1)\cdot d\big)$ for the discriminator and is $O\big(kd\big)$ for the generator, where k is the number of hidden units in generator's MLP. The computational complexity mainly involves the matrix multiplication operations. Then, the computational complexity per epoch is $O\big((2n_p+n_u)\cdot(M+N)\cdot d^2\big)$ for the discriminator, and $O\big(n_u k d^2\big)$ for the generator.

5 EXPERIMENTAL RESULTS

In this section, we aim to answer the following research questions:

RQ1: Can the proposed PURE model outperform the state-of-the-art recommendation methods?

RQ2: What is the parameter sensitivity for PURE in terms of positive prior π_p and the generator's random noise input magnitude δ? Does pre-train affect the ranking performance?

RQ3: How does the running time of PURE compare with other baselines?

Dataset	# Users	# Items	# Interactions	Sparsity
Movielens	6,040	3,706	1,000,209	4.46%
Yelp	25,677	25,815	731,671	0.11%

Table 1: Statistics of the data sets

5.1 Experimental Settings

5.1.1 Data sets. We conduct the experiments on two publicly accessible data sets: Movielens[2] and Yelp[3]. For Yelp data set, due to the sparsity of the ratings among the data, we adopt the preprocessing step from [15] by keeping the users with more than 10 item interactions. To perform implicit recommendation, following the experimental setting of [40], only the 4-star and 5-star ratings in these data sets are treated as positive feedback, and the rest are unknown feedback. In this way, the data is transformed into the user-item interaction matrix $\mathcal{R}$ with each entry being either 0 or 1. The details of these data sets are summarized in Table 1.

5.1.2 Evaluation Protocol. To evaluate the performance of all methods, we adopt the official 80%|20% random split and perform sampled evaluation [15, 22, 24] to speed up the computation. In the evaluation stage, only a smaller set of random items is used as the candidates pool for ranking predictions. Due to the fact that Movielens and Yelp have been pre-processed to only keep the users with at least 20 or 10 relevant items, we adopt the random leave-ten-out (for Movielens) and leave-five-out (for Yelp) strategy to split them into the train set and the test set. Unlike the leave-one-out sampled metric strategy being used in [15, 22] which has candidates pool of size 100 and it may introduce bias into evaluation results. We follow the suggestion of [24] and make the candidates pool with a larger size of 500 items. This is a good trade-off pool size for the sampled evaluation where both computation cost and true performance consistency are well balanced. The eventual performance of the predicted ranked list is evaluated by Precision (P@k), Normalized Discounted Cumulative Gain (NDCG@k), where $k=\{3,5,10\}$, Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR). Note that we have modified the evaluation protocols significantly to couple with the sampled evaluation and actual needs (more than one recommended items per user) in real applications, the results are not directly comparable with previous ones [4, 15, 22, 40]. Nevertheless, we have utilized a rich bundle of popular baselines and all have been evaluated using the same evaluation protocols.

5.1.3 Baselines. We considered five categories of recommendation methods for comparisons:

- *Traditional collaborative filtering*: **ItemPop** is a non-personalized method that recommend the most popular items to each user. **SlopeOne** [25] is also another item-based collaborative filtering method; **Co-clustering** [11] identifies overlapping co-clusters of users & items and infers relevance score using cluster statistics.
- *Traditional matrix factorization*: **SVD** [23] infers the user-item interaction score as the sum of user bias, item bias, and the product of user & item latent factors; **NMF** [28] is similar to SVD, but the user and item factors are computed under a non-negative constraints; **PMF** [35] infers the interaction score from user and item probabilistic latent factors with Frobenius regularization.
- *Neural collaborative filtering*: **BPR** [34] learns the user and item embeddings using user-specific pairwise preferences between a pair of items; **LambdaFM** [42] learns the embeddings using pairwise ranking loss along with lambda surrogate; **GMF** [15] learns the embeddings using pointwise label information along with relation mapping embedding.
- *GAN based recommenders*: **GraphGAN** [38] builds a discriminator to predict the connectivity of user & item, and a generator to learn the joint discrete distribution; **IRGAN** [40] builds a discriminator using matrix factorization, and a generator to extract relevant items using policy gradient; **CFGAN** [4] uses a generator to generate the purchase vectors for users, and a discriminator to differentiate the real purchase vectors and the fake ones.
- *PU-learning based recommenders*: **PU-GMF** [21] modifies the PN learning objective of GMF with its PU learning version, and feed the model with positive data and sampled unlabeled data; **PURE** is our proposed model.

5.1.4 Reproducible settings. To guarantee a fair comparison between all baselines, we fix the embedding size d as 8 and 16 for all models on Movielens and Yelp, respectively. Meanwhile, the input allowed for all models would be the rating matrix $\mathcal{R}$ only, no side information or additional features are supplied. All models are validated on the performance of P@5. The learning rate is searched from $\{1\times10^{-4}, 1\times10^{-3}, 1\times10^{-2}\}$, the positive class prior π_p is searched from $\{1\times10^{-6}, 1\times10^{-5}, 1\times10^{-4}, 1\times10^{-3}, 1\times10^{-2}, 1\times10^{-1}\}$, the generator's input noise magnitude δ is

[1]The negative sampling ratio for PN learning is set as $C=1$ in the algorithm.
[2]https://grouplens.org/datasets/movielens/
[3]https://github.com/hexiangnan/sigir16-eals/tree/master/data

Movielens-1m	P@3	P@5	P@10	NDCG@3	NDCG@5	NDCG@10	MAP	MRR
ItemPop	0.2805	0.2400	0.1845	0.2961	0.2725	0.2883	0.2371	0.5038
SlopeOne[25]	0.3954	0.3736	0.3124	0.3887	0.3775	0.3453	0.2958	0.4981
Co-clustering[11]	0.4826	0.4612	0.4195	0.4533	0.4475	0.4283	0.3500	0.5105
SVD[23]	0.4187	0.3563	0.2621	0.4483	0.4107	0.4224	0.3546	0.6680
NMF[28]	0.5262	0.4916	0.4118	0.5238	0.5034	0.4603	0.4002	0.6279
PMF[35]	0.4108	0.3975	0.3633	0.3819	0.3817	0.3678	0.3182	0.4406
BPR[34]	0.6604	0.7379	0.8272	0.6339	0.6930	0.7710	0.3793	0.6714
LambdaFM[42]	0.6365	0.7116	0.8072	0.6070	0.6669	0.7493	**0.9516**	0.6488
GMF[15]	0.6546	0.7284	0.8156	0.6254	0.6798	0.7583	0.8354	0.6594
GraphGAN[38]	0.4731	0.5538	0.5209	0.4433	0.5072	0.5019	0.4198	0.4998
IRGAN[40]	0.2732	0.2326	0.1774	0.2885	0.2644	0.2779	0.2279	0.4926
CFGAN [4]	0.6209	0.6978	0.7983	0.5902	0.6517	0.7379	0.8114	0.6337
PU-GMF[15]+[21]	0.6639	0.7394	0.8268	0.6398	0.6963	0.7724	0.8639	0.6762
PURE (ours)	**0.6824**	**0.7523**	**0.8351**	**0.6532**	**0.7094**	**0.7829**	0.8703	**0.6895**

Table 2: Evaluation results of Movielens data set

Yelp	P@3	P@5	P@10	NDCG@3	NDCG@5	NDCG@10	MAP	MRR
ItemPop	0.1335	0.1124	0.0842	0.1489	0.1592	0.1976	0.1596	0.2858
SlopeOne[25]	0.2053	0.1917	0.1567	0.2008	0.1983	0.2122	0.1986	0.2869
Co-clustering[11]	0.2216	0.1929	0.1475	0.2397	0.2499	0.2937	0.2423	0.3913
SVD[23]	0.2635	0.2157	0.1527	0.2960	0.3083	0.3693	0.2981	0.4880
NMF[28]	0.3788	0.3474	0.2767	0.3754	0.3652	0.3820	0.3560	0.4781
PMF[35]	0.2772	0.2750	0.2478	0.2564	0.2631	0.2671	0.2573	0.3132
BPR[34]	0.4634	0.5423	0.6561	0.4345	0.4968	0.5918	0.5797	0.4910
LambdaFM[42]	0.3920	0.4653	0.5757	0.3659	0.4236	0.5149	0.8173	0.4242
GMF[15]	0.4416	0.5230	0.6426	0.4122	0.4764	0.5758	0.6556	0.4715
GraphGAN[38]	–	–	–	–	–	–	–	–
IRGAN[40]	0.1276	0.1081	0.0816	0.1424	0.1525	0.1901	0.1551	0.1735
CFGAN [4]	0.2309	0.2824	0.3699	0.2140	0.2541	0.3247	0.3260	0.2676
PU-GMF[15]+[21]	0.4866	0.5666	0.6800	0.4560	0.5196	0.6149	0.7857	0.5102
PURE (ours)	**0.5038**	**0.5830**	**0.6935**	**0.4736**	**0.5365**	**0.6297**	**0.9206**	**0.5264**

Table 3: Evaluation results of Yelp data set

searched on $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3\}$. Training is accelerated with pre-train, where we initialize PURE's generator with random weights and initialize the discriminator with PU-GMF's embedding weights.

5.2 Performance Comparison (RQ1)

From the evaluation results shown in Table 2 and Table 3, we observe that PURE achieves the best performance on most metrics. The neural collaborative filtering methods and GAN-based methods are usually very competitive overall. For the Movielens data set, as we can see in Table 2, BPR, LambdaFM, PU-GMF, and CFGAN also perform relatively well in terms of P@k and NDCG@k comparing with other baselines. One interesting observation is that LambdaFM has very high values on the MAP metric. That is because pairwise learning is position-independent and pairwise-ordering at the bottom of the ranking list would impact the learning loss as much as the top pairs. Meanwhile, LambdaFM is particularly designed for optimizing the overall ranking performance. We follow the setting from the papers of BPR and IRGAN to tune the learning rate, number of epochs, etc. We found that BPR is very sensitive to the sample sequence in the training batches, and IRGAN's performance is largely impacted by its hyperparameters and pretraining. We have performed a comprehensive model tuning in a reasonable time period for all baselines and reported their best performance for a fair comparison.

For the results of the Yelp data set, the best frameworks are PU-GMF and PURE, followed by neural collaborative filtering methods such as BPR, LambdaFM, and GMF. The GraphGAN method fails to finish training on Yelp since it needs to compute the graph softmax and generate a huge amount of neighbor vertices for each existing vertex. The traditional collaborative filtering and MF-based methods do not have a satisfactory performance on Yelp. We also observe that GAN-based methods could easily fail to converge even with careful hyperparameter tuning. The reasons for their poor performance are two-fold: First, these methods are taking the unobserved data as negative samples without the negative sampling procedure, resulting in an unbalanced training data problem, especially for Yelp data set, which is much sparser than the other two data sets. Second, they didn't use continuous space sampling, and generating with discrete sampling will end up with poor model expressiveness especially when dealing with large-scale sparse data set. As a comparison, PU learning will help sample from both the observed and unobserved entries and the generator will further learn the data distribution and generate continuous user-item embeddings to increase the model expressiveness. Similarly, the models with pairwise loss (BPR and LambdaFM) also perform relatively well on MAP due to their position-independent properties in the modeling.

5.3 Parameter Study (RQ2)

Regarding the hyper-parameters sensitivity in PURE, we show the performance of P@5, NDCG@5, MAP, and MRR with respect to the positive class prior π_p and the magnitude of generator's input noise δ. We perform this parameter study on a smaller version of Movielens data set that has 100k reviews, i.e., Movielens-100k,

	With Pretrain		Without Pretrain	
	P@5	NDCG@5	P@5	NDCG@5
Movielens	**0.7523**	**0.7094**	0.7101	0.6568
Yelp	**0.5830**	**0.5365**	0.5340	0.4864

Table 4: Performance of PURE with/without pre-training.

because of its small size so that we can well-tune all the competitive baselines with a reasonable amount of effort. Note that we adopted the full test set evaluation instead of sampled evaluation, however, since Movielens-100k's default train|test split assigns only half of users from train into the test set, the final evaluation performance is much lower than Movielens-1m with sampled evaluation.

First, we can see that PURE achieves the best performance with carefully selected hyper-parameters. We observe in Figure 2 (In Appendix) that PURE outperforms (on average 1.5%) all the competitors when π_p is set to 0.0001. Meanwhile, we see that PURE has a performance guarantee if π_p falls into the range of $[0.00001, 0.001]$ which means the underlying true density of the positive samples is very sparse. Namely, each user would only show interest in a very small number of items on average, which is reasonable in real-world applications. Second, Figure 3 (In Appendix) shows that PURE is not very sensitive to the conditional noise magnitude. Starting from $\delta = 0.005$ to $\delta = 0.1$, we observe that PURE can almost outperform every baseline in all metrics. It is because the generator could produce high-quality fake embeddings to help improve the discriminating ability of the discriminator. We conjecture that fine-tuning the structure of MLP layers could further improve the expressiveness of the generator, which in turn improves the overall performance. The exploration of the optimal model structure is left for future work. Third, to demonstrate the utility of pre-training, we compared the performance of two different versions for PURE - with and without pre-training. For PURE without pre-training, we initialize the embedding layers of the discriminator and the MLP layers of the generator with random weights. For PURE with pre-train, we first train a PU-GMF model, and then assign its embedding weights to PURE's discriminator, but the generator is still random initialized. As shown in Table. 4, the relative improvements of utilizing pre-training are roughly 1%, 4%, and 5% for Movielens-100k, Movielens-1m, and Yelp, respectively. We empirically observe that PURE with pre-training converges faster with less training epochs. Both above observations justify the usefulness and efficiency of our proposed pre-training method for initializing PURE.

5.4 Running Time (RQ3)

In Figure 4 (In Appendix), we compare the running time between PURE and other baselines on Movielens-1m. The circle size represents the average performance (NDCG@5 in this case) of the corresponding method and the x-axis records their running time in log scale. As we can see, the traditional matrix factorization methods run very fast but with limited performance. PURE performs the best but a little slower than PU-GMF due to the extra training time of the generator. The pairwise loss based models, such as BPR and LambdaFM, are comparable in terms of performance. CFGAN performs well on this data set since it also performs continuous sampling. IRGAN suffers the issue of high computational complexity due to multiple reasons, e.g., dynamic negative sampling, softmax operations, high generator, and discriminator epochs. GraphGAN and SlopeOne need to loop over all users and items multiple times and therefore, have the highest computational complexity.

6 RELATED WORK

6.1 Recommender Systems

Algorithms and frameworks regarding recommendation systems have been widely studied in recent years due to their business success to attract traffic or improve profit in different domains [1, 7, 19, 27, 32, 36, 39, 47]. Collaborative filtering based methods play an important role in recommender systems and gain major attention [23] for recent decades. Within collaborative filtering, latent factor or embedding based algorithms such as matrix factorization [23], factorization machines [33] and their variants [28, 35, 42] have been successfully applied in recommender systems. With the development of deep neural networks, deep learning based recommender systems become a hot research topic since they introduce non-linearity and increase model expressiveness [44]. Traditional matrix factorization based algorithms have been transformed into their deep model versions, such as neural collaborative filtering [15] and deep factorization machine [13]. In order to approach the true distribution of the user and items, generative adversarial networks have been adopted for information retrieval [18, 40], network mining [10, 38, 43, 46], and heterogeneous learning [45, 48, 49]. IRGAN [40] formulates a minimax game where the generator learns the discrete relevance distribution of users and items for synthesizing the indistinguishably fake user-item tuples while the discriminator identifies whether one user-item tuple is real or not. Following this idea, GraphGAN [38] learns the underlying connection distribution over vertices in an adversarial framework for graph representation learning. HeGAN [18] further proposed the relation-aware generator and discriminator to encode the heterogeneous information network with multiple types of vertices and edges. All these methods take the unlabeled user and item interactions as negative samples, which is a non-valid assumption for real-world applications. In this paper, we addressed this problem by utilizing the GAN-based retrieval model and train its discriminator under the PU learning framework.

6.2 Positive Unlabeled Learning

PU learning [14, 16, 21, 30] is a variant of the classical PN learning, where the training data only consists of positive and unlabeled samples. This learning setting fits with the applications that do not require fully supervised data. The pioneering work [5, 26] of PU learning was initialized two decades ago. and the state-of-the-art PU learning approaches are mainly focused on unbiased PU risk estimators. Starting from [9], which treats the unlabeled data as a weighted mixture of positive and negative data and has an unbiased estimator if positive and negative conditional densities are disjoint, multiple variants [8, 21, 30] have been proposed. It has been proven in [8] that an unbiased PU estimator can be learned if the loss is symmetric. Later on, the analysis in [30] shows that the unbiased estimator could be convex for loss functions that meet the linear-odd condition. However, the aforementioned approaches are not applicable to very flexible models, where the overall risk of the estimator will become negative. [21] has shown that by imposing a non-negative operator on the estimated empirical risk term of the unlabeled data, the non-negative risk estimator will reduce

the overfitting phenomenon, which opens the door for adopting deep neural networks into PU learning frameworks. PU learning has not been extensively explored on recommender systems, even inherently the recommendation problem fits the PU learning scenario very well. Most related work includes PU learning for matrix completion [17] and one-class collaborative filtering [31]. Our work is different from these methods in that, PURE does not impose any assumption on the distribution of users and items, and employs the GAN framework to learn the real distribution of the user-item interaction in a continuous embedding space.

7 CONCLUSION

In this paper, we proposed a novel recommendation framework based on the generative adversarial network. Its discriminator is trained using PU learning with an unbiased risk estimator, while the generator learns the underlying continuous distribution of users and items to generate high-quality fake embeddings of them. We theoretically analyzed the performance of PURE from multiple aspects, and performed extensive experiments to demonstrate its effectiveness and efficiency for personalized ranking problems in comparison with a rich set of strong baselines.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation under Award No. IIS-1947203 and IIS-2002540, and Agriculture and Food Research Initiative (AFRI) grant no. 2020-67021-32799/project accession no.1024178 from the USDA National Institute of Food and Agriculture. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] Yikun Ban and Jingrui He. 2021. Local Clustering in Contextual Multi-Armed Bandits. In *WWW*.
[2] Jessa Bekker and Jesse Davis. 2018. Learning From Positive and Unlabeled Data: A Survey. *CoRR* (2018). arXiv:1811.04820 http://arxiv.org/abs/1811.04820
[3] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *ICML*. 89–96.
[4] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks. In *CIKM*. 137–146.
[5] Francesco De Comité, François Denis, Rémi Gilleron, and Fabien Letouzey. 1999. Positive and Unlabeled Examples Help Learning. In *ALT*. 219–230.
[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. 191–198.
[7] Boxin Du, Lihui Liu, and Hanghang Tong. 2021. Sylvester Tensor Equation for Multi-Way Association. In *KDD*. ACM.
[8] Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. 2014. Analysis of Learning from Positive and Unlabeled Data. In *NeurIPS*. 703–711.
[9] Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *KDD*. 213–220.
[10] Dongqi Fu, Zhe Xu, Bo Li, Hanghang Tong, and Jingrui He. 2020. A View-Adversarial Framework for Multi-View Network Embedding. In *CIKM*.
[11] Thomas George and Srujana Merugu. 2005. A Scalable Collaborative Filtering Framework Based on Co-Clustering. In *ICDM*. 625–628.
[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NeurIPS*. 2672–2680.
[13] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *IJCAI*. 1725–1731.
[14] Tianyu Guo, Chang Xu, Jiajun Huang, Yunhe Wang, Boxin Shi, Chao Xu, and Dacheng Tao. 2020. On Positive-Unlabeled Classification in GAN. In *CVPR*. 8382–8390.
[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[16] Ming Hou, Brahim Chaib-draa, Chao Li, and Qibin Zhao. 2018. Generative Adversarial Positive-Unlabelled Learning. In *IJCAI*. 2255–2261.
[17] Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit S Dhillon. 2015. PU Learning for Matrix Completion. In *ICML*. 2445–2453.
[18] Binbin Hu, Yuan Fang, and Chuan Shi. 2019. Adversarial Learning on Heterogeneous Information Networks. In *KDD*. 120–129.
[19] Dietmar Jannach and Michael Jugovac. 2019. Measuring the Business Value of Recommender Systems. *ACM TMIS* 10, 4 (2019).
[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[21] Ryuichi Kiryo, Gang Niu, Marthinus Christoffel du Plessis, and Masashi Sugiyama. 2017. Positive-Unlabeled Learning with Non-Negative Risk Estimator. In *NeurIPS*. 1675–1685.
[22] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.
[23] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.
[24] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *KDD*. 1748–1757.
[25] Daniel Lemire and Anna Maclachlan. 2005. Slope One Predictors for Online Rating-Based Collaborative Filtering. In *SDM*. 471–475.
[26] Fabien Letouzey, François Denis, and Rémi Gilleron. 2000. Learning from Positive and Unlabeled Examples. In *ALT*. 71–85.
[27] Xu Liu, Jingrui He, Sam Duddy, and Liz O'Sullivan. 2019. Convolution-Consistent Collective Matrix Completion. In *CIKM*. 2209–2212.
[28] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. 2014. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems. *IEEE TII* 10, 2 (2014), 1273–1284.
[29] Larry M. Manevitz and Malik Yousef. 2001. One-Class SVMs for Document Classification. *Journal of Machine Learning Research* 2 (2001), 139–154.
[30] Gang Niu, Marthinus Christoffel du Plessis, Tomoya Sakai, Yao Ma, and Masashi Sugiyama. 2016. Theoretical Comparisons of Positive-Unlabeled Learning against Positive-Negative Learning. In *NeurIPS*. 1199–1207.
[31] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. [n.d.]. One-Class Collaborative Filtering. In *ICDM*.
[32] Dae Hoon Park and Yi Chang. 2019. Adversarial Sampling and Training for Semi-Supervised Information Retrieval. In *WWW*. 1443–1453.
[33] Steffen Rendle. 2010. Factorization Machines. In *ICDM*. 995–1000.
[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*.
[35] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *NeurIPS*. 1257–1264.
[36] Huajie Shao, Dachun Sun, Jiahao Wu, Zecheng Zhang, Shuochao Yao, Shengzhong Liu, Tianshi Wang, Chao Zhang, and Tarek F. Abdelzaher. 2020. paper2repo: GitHub Repository Recommendation for Academic Papers. In *WWW*. 629–639.
[37] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. In *KDD*. 2309–2318.
[38] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning With Generative Adversarial Nets. In *AAAI*. 2508–2515.
[39] Haonan Wang, Chang Zhou, Hongxia Yang, Carl Yang, and Jingrui He. 2021. Controllable Gradient Item Retrieval. In *WWW*.
[40] 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 *SIGIR*. 515–524.
[41] Da Xu, Chuanwei Ruan, Jason Cho, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Knowledge-Aware Complementary Product Representation Learning. In *WSDM*. 681–689.
[42] 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 *CIKM*. 227–236.
[43] Si Zhang, Hanghang Tong, Yinglong Xia, Liang Xiong, and Jiejun Xu. 2020. NetTrans: Neural Cross-Network Transformation. In *KDD*. 986–996.
[44] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *Comput. Surveys* 52, 1 (2019), 38.
[45] Lecheng Zheng, Yu Cheng, Hongxia Yang, Nan Cao, and Jingrui He. 2021. Deep Co-Attention Network for Multi-View Subspace Learning. *CoRR* abs/2102.07751 (2021).
[46] Dawei Zhou, Lecheng Zheng, Jiejun Xu, and Jingrui He. 2019. Misc-GAN: A Multi-scale Generative Model for Graphs. *Frontiers Big Data* 2 (2019), 3.
[47] Yao Zhou, Fenglong Ma, Jing Gao, and Jingrui He. 2019. Optimizing the Wisdom of the Crowd: Inference, Learning, and Teaching. In *KDD*. 3231–3232.
[48] Yao Zhou, Lei Ying, and Jingrui He. 2017. MultiC^2: an Optimization Framework for Learning from Task and Worker Dual Heterogeneity. In *SDM*. 579–587.
[49] Yao Zhou, Lei Ying, and Jingrui He. 2019. Multi-task Crowdsourcing via an Optimization Framework. *ACM TKDD* 13, 3 (2019), 27:1–27:26.

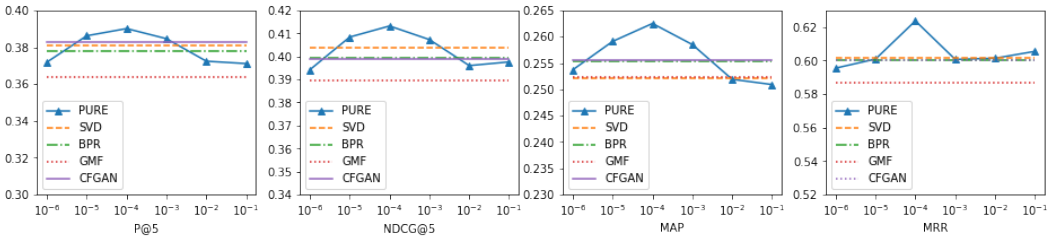

Figure 2: Positive class prior π_p

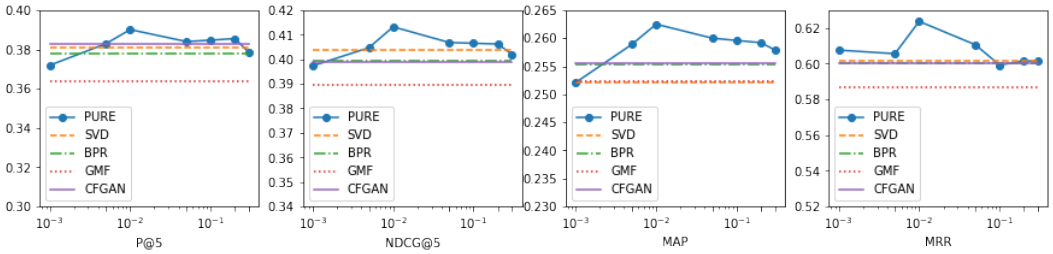

Figure 3: Generator's input noise magnitude δ

A APPENDIX

In all the following proofs, we denote each user-item tuple (u, i) in recommendation as one data sample $\boldsymbol{x}$ for simplicity.

A.1 Proof of Theorem 1

Theorem 1 states that the Estimation Error Bound of PU learning is tighter than that of PN learning if and only if:

$$n_u \geq \frac{\sqrt{C}\, n_p}{\left(1 - \left(\sqrt{C}+1\right)\pi_p\right)^2}$$

Proof. The differences of PN learning and PU learning in terms of the EER bounds in Lemma 1 reflect the differences w.r.t. their risk minimizers. We define:

$$\alpha_{pu,pn} := \frac{\pi_p/\sqrt{n_p} + 1/\sqrt{n_u}}{(1-\pi_p)/\sqrt{n_n}} \tag{18}$$

For simplicity, let's denote $\rho_{pu} := n_p/n_u$ and we know $\rho_{pn} := n_p/n_n = 1/C$. Then, by setting $\alpha_{pu,pn} \leq 1$, we have:

$$\pi_p + \sqrt{\rho_{pu}} \leq \frac{1}{\sqrt{C}}(1-\pi_p) \Leftrightarrow \rho_{pu} \leq \frac{1}{\sqrt{C}}\left(1 - \left(\sqrt{C}+1\right)\pi_p\right)^2 \tag{19}$$

It is rather straightforward to get the conclusion in Eq. (A.1) by solving the above inequality. □

Lemma 1. *[**Estimation Error Bound (EEB)**] Let $\mathcal{F}$ be the function class, and $\hat{f}_{pn}$ and $\hat{f}_{pu}$ be the empirical risk minimizer of $\hat{R}_{pn}(D)$ and $\hat{R}_{pu}(D)$ for discriminator D that belongs to PN learning and PU learning, respectively. Then, the EEB of $\hat{f}_{pu}$ is tighter than $\hat{f}_{pn}$ with probability at least $1-\delta$ when:*

$$\frac{\pi_p}{\sqrt{n_p}} + \frac{1}{\sqrt{n_u}} < \frac{\pi_n}{\sqrt{n_n}} \tag{20}$$

if the loss L is symmetric and Lipschitz continuous, and the Rademacher complexity of $\mathcal{F}$ decays in $O(1/\sqrt{n})$ for data of size n drawn from $p_{data}(x)$, $p_p(x)$, and $p_n(x)$.

Proof can be referred to [30] for details. Based on the above theorem, we know that PU learning is highly likely to outperform PN learning when Eq. (20) and certain mild conditions [30] are satisfied.

A.2 Proof of Proposition 1

Proposition 1 states that when the generator G is fixed, the optimal discriminator D is:

$$D^*(x) = \frac{\pi_p p_p(x)}{p_u(x) + p_g(x)}$$

Proof. We know that the underlying true data distribution is: $p_{data}(x) = \pi_p p_p(x) + (1-\pi_p)p_n(x)$. Furthermore, we also denote the generator's output distribution as $p_g(x)$. Then, the objective of the discriminator D is as follows for fixed G:

$$\begin{aligned}
&\max_D V(D) \\
=&\pi_p \int_x p_p(x)\log(D(x))dx - \pi_p \int_x p_p(x)\log(1-D(x))dx \\
&+ \int_x p_u(x)\log(1-D(x))dx + \int_z p_z(z)\log(1-D(G(z)))dz \\
=&\int_x \Big(-\pi_p \cdot p_p(x) + p_u(x) + p_g(x)\Big)\log(1-D(x))dx \\
&+ \pi_p \int_x p_p(x)\log(D(x))dx \\
=&\int_x \Big((1-\pi_p)\cdot p_n(x) + p_g(x)\Big)\log(1-D(x))dx \\
&+ \int_x \pi_p \cdot p_p(x)\log(D(x))dx
\end{aligned} \tag{21}$$

Here, we assume that the unlabeled distribution can also be decomposed as $p_u(x) \approx \pi_p p_p(x) + (1-\pi_p)p_p(x)$ approximately since the sampled positive tuples are extremely sparse in the overall user-item population. Next, we know that for the problem of $\max_y a\log(y) + b\log(1-y)$, it achieves the optimal value [12] when

	latent dim. d	batch size	learning rate	# epoch	pos. prior π_p	noise mag. δ
Movielens-100k	5	128	0.001	100	0.0001	0.01
Movielens-1m	8	128	0.001	100	0.00001	0.01
Yelp	16	512	0.001	200	0.000001	0.01

Table 5: Reproducible parameter setting

$y^* = \frac{a}{a+b}$. Let $a = \pi_p \cdot p_p(x)$ and $b = (1-\pi_p) \cdot p_n(x) + \cdot p_g(x)$,

$$D^* = \frac{\pi_p \cdot p_p(x)}{\pi_p \cdot p_p(x) + (1-\pi_p) \cdot p_n(x) + p_g(x)} = \frac{\pi_p \cdot p_p(x)}{p_u(x) + p_g(x)}$$

□

A.3 Proof of Proposition 2

Theorem 2 states that when the discriminator D fixed, the optimization of the generator G is equivalent to minimize: $-2H\left(\frac{\pi_p}{2}\right) + \pi_p \cdot \mathrm{KL}\left(p_p(x) \middle\| \frac{p_u(x)+p_g(x)}{2}\right) + (2-\pi_p) \cdot \mathrm{KL}\left(\frac{(1-\pi_p)p_n(x)+p_g(x)}{2-\pi_p} \middle\| \frac{p_u(x)+p_g(x)}{2}\right)$.

Proof. For fixed optimal discriminator, we substitute D^* into the objective of Eq. (21) and have the following objective:

$$\begin{aligned}
&\min V(G) \\
=&\pi_p \int_x p_p(x)\log(D(x))dx - \pi_p \int_x p_p(x)\log(1-D(x))dx \\
&+ \int_x p_u(x)\log(1-D(x))dx + \int_z p_z(z)\log(1-D(G(z)))dz \\
=&\int_x \pi_p p_p(x)\log(D(x))dx \\
&+ \int_x \Big((1-\pi_p)p_n(x) + p_g(x)\Big)\log(1-D(x))dx \\
=&\int_x \Big((1-\pi_p)p_n(x) + p_g(x)\Big)\log\left(1 - \frac{\pi_p p_p(x)}{p_u(x)+p_g(x)}\right)dx \\
&+ \int_x \pi_p p_p(x)\log\left(\frac{\pi_p p_p(x)}{p_u(x)+p_g(x)}\right)dx \\
=&\int_x \pi_p p_p(x)\left[\log\frac{\pi_p}{2} + \log\left(\frac{p_p(x)}{\frac{p_u(x)+p_g(x)}{2}}\right)\right]dx + (2-\pi_p)\cdot \\
&\int_x \frac{(1-\pi_p)p_n(x)+p_g(x)}{2-\pi_p}\left[\log\left(\frac{\frac{(1-\pi_p)p_n(x)+p_g(x)}{2-\pi_p}}{\frac{p_u(x)+p_g(x)}{2}}\right) + \log\frac{2-\pi_p}{2}\right]dx \\
=&-2\mathrm{H}\left(\frac{\pi_p}{2}\right) + \pi_p \cdot \mathrm{KL}\left(p_p(x)\middle\|\frac{p_u(x)+p_g(x)}{2}\right) \\
&+ (2-\pi_p)\cdot \mathrm{KL}\left(\frac{(1-\pi_p)p_n(x)+p_g(x)}{2-\pi_p}\middle\|\frac{p_u(x)+p_g(x)}{2}\right)
\end{aligned}$$

□

A.4 Proof of Theorem 2

Theorem 2 states that the global minimum could be achieved if and only if $p_p(x) = \frac{p_u(x)+p_g(x)}{2}$. At that point, the objective value of the framework $V(G,D)$ converges to $-2\mathrm{H}\left(\frac{\pi_p}{2}\right)$, and the value of $D(x)$ reaches $\frac{\pi_p}{2}$.

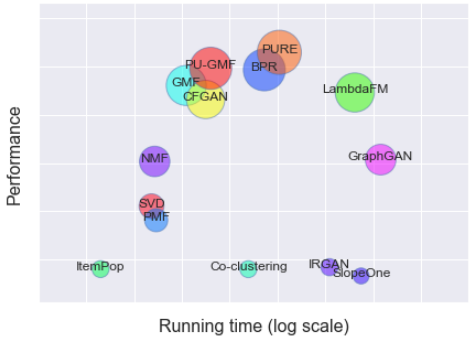

Figure 4: Running time of baselines on Movielens-1m

Proof. From Proposition 2, we can directly get the minimum of the optimal generator as $-2\mathrm{H}(\frac{\pi_p}{2})$ if and only if these three distributions are identical: $p_p(x) = \frac{p_u(x)+p_g(x)}{2}$ and $\frac{(1-\pi_p)p_n(x)+p_g(x)}{2-\pi_p} = \frac{p_u(x)+p_g(x)}{2}$. By solving the second equality, we have the following equality equivalences:

$$\begin{aligned}
\frac{p_u(x)+p_g(x)}{2} &= \frac{(1-\pi_p)p_n(x)+p_g(x)}{2-\pi_p} \\
(2-\pi_p)p_u(x) + (2-\pi_p)p_g(x) &= (2-2\pi_p)p_n(x) + 2p_g(x) \\
(2-\pi_p)p_u(x) - \pi_p p_g(x) &= (2-2\pi_p)p_n(x) \\
(2-\pi_p)p_u(x) - \pi_p p_g(x) &= 2p_u(x) - 2\pi_p p_p(x) \\
p_p(x) &= \frac{p_u(x)+p_g(x)}{2}
\end{aligned} \tag{22}$$

Here, using the last formula in Eq. (22), we show that the two equalities of Theorem 2 are exactly the same. Easily, if we substitute either of them into the optimal D^*, we will always have $D^*(x) = \frac{\pi_p}{2}$.

□

A.5 Reproducible Setting

To recover the experimental results, below are the required reproducible settings: For all three data sets, we trained the generator from scratch with "lecun_uniform" random initialization on the MLP layers. For discriminator, we initialize its user and item embedding weights with a pre-trained PU-GMF weights. The model losses for both discriminator and generator are binary cross entropy loss and they are optimizer using Adam optimizer. The local epochs for the discriminator and the generator are 1 and 10, respectively. The MLP layer in generator has `ReLU` activation which has been verified to perform better than other activation functions, such as `LeakyReLU`, `Sigmoid`, `Linear`, etc. Other detailed hyperparameter settings including user & item embedding dimension, training batch size, learning rate, number of training epochs, positive class prior, as well as the noise input magnitude for generators are summarized in Table 5.