CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks

Dong-Kyu Chae
Hanyang University
Seoul, Korea
kyu899@hanyang.ac.kr

Jin-Soo Kang
Hanyang University
Seoul, Korea
jensoo7023@hanyang.ac.kr

Sang-Wook Kim*
Hanyang University
Seoul, Korea
wook@hanyang.ac.kr

Jung-Tae Lee
NAVER Corporation
SeongNam, Korea
jungtae.lee@navercorp.com

ABSTRACT

Generative Adversarial Networks (GAN) have achieved big success in various domains such as image generation, music generation, and natural language generation. In this paper, we propose a novel GAN-based collaborative filtering (CF) framework to provide higher accuracy in recommendation. We first identify a fundamental problem of existing GAN-based methods in CF and highlight it quantitatively via a series of experiments. Next, we suggest a new direction of *vector-wise adversarial training* to solve the problem and propose our GAN-based CF framework, called CFGAN, based on the direction. We identify a unique challenge that arises when vector-wise adversarial training is employed in CF. We then propose three CF methods realized on top of our CFGAN that are able to address the challenge. Finally, via extensive experiments on real-world datasets, we validate that vector-wise adversarial training employed in CFGAN is really effective to solve the problem of existing GAN-based CF methods. Furthermore, we demonstrate that our proposed CF methods on CFGAN provide recommendation accuracy consistently and universally higher than those of the state-of-the-art recommenders.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Top-N recommendation, collaborative filtering, generative adversarial networks, implicit feedback

ACM Reference Format:
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 *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271743

*Corresponding author

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271743

1 INTRODUCTION

The *collaborative filtering* (CF) is one of the most popular and widely used methods in recommender systems [1]. It aims at providing a personalized list of items to a user by collecting her prior preference history [25]. Users' preference histories are usually represented in a form of a sparse matrix where a column indicates a user, a row indicates an item, and a value does a user's preference on an item. Here, a user's preference (or feedback) can be *explicit*, provided mostly in ratings from 1 to 5, or *implicit*, provided typically in a unary fashion such as purchases or clicks [38]. This paper focuses on CF with implicit feedback since this form of data is more easier to collect [20, 31], and thus has received much attention in the research community [13]. Throughout the paper, we will use the term *purchase* to denote any kind of implicit feedback, unless otherwise stated.

Among a number of CF methods, *model-based CF* methods, which build a model to learn the past preference patterns of users on items, are known the most successful [30]. Some of the most popular model-based CF methods are based on *matrix factorization* (MF), which learn the *linear interactions* between latent features of users and items. Their prominent examples include BiasedMF [21], SVD++ [20], and PMF [29] on the rating prediction task, and WRMF [15], BPR [34], PureSVD [7], SLIM [30], CLiMF [36], and FISM [18] on the top-N recommendation task. On the other hand, deep neural network (DNN) based CF methods have lately gained increasing attention [46]. Owing to the DNN's capability of discovering the *non-linear interactions* of users and items' latent factors and approximating an arbitrary continuous function [14, 22], DNN-based CF models have also impressive achievements on both rating prediction (e.g., AutoRec [35] and CDL [40]) and top-N recommendation tasks (e.g., CDAE [44] and NCF [13]).

Recently, Goodfellow et al. [11] proposed *Generative Adversarial Networks* (GAN), one of variants of DNNs, which learns to mimic any distribution of given data. GAN consists of two models: one is a *generative model* (shortly, G) and the other is a *discriminative model* (shortly, D). During learning, G tries to generate realistic (but fake in reality) data and passes it to D; D evaluates the possibility that the data came from the ground truth (i.e., real data) rather than

from G [5]. The training procedure is a minimax game between G and D: G tries to increase the error rate of (i.e., fool) D while D tries to accurately discriminate real data from the fake ones. As a result of this adversarial process, G learns how to generate realistic data, e.g., photographs that look at least superficially authentic to human eyes. GAN has already gained huge success in image generation [6, 32], and has also received much interest in other tasks such as natural language generation [45] and music generation [8].

However, GAN has not gained much attention yet in the field of recommender systems. IRGAN [42] and GraphGAN [41] are the pioneering methods that demonstrated GAN's potential for the success in CF. Their main idea is, given a user, to let G generate (more precisely, *sample*) items that the user might purchase, and to let D discriminate a user's ground truth items (i.e., her purchased items in the past) from those sampled by G, with the expectation that G will eventually capture the true distribution of users' preferences over items [42]. Since G generates a *single item index* each time, this discontinuity makes it difficult to update the parameters of G properly via standard back-propagation [45]. Hence, the *policy-gradient based reinforcement learning* [43] is employed to guide G by using D's output as a reward signal to G. The competition between G and D drives both of them to help each other to achieve their own goals; finally, G can sample the indices of items that are plausible and of high quality for recommendation [41, 42]. Empirically, IRGAN and GraphGAN have achieved a promising accuracy in recommendation on real-world data.

In this paper, we point out the problem in the above GAN-based CF methods, and suggest a new direction to address this problem. We claim that the advantage of the adversarial training is not fully exploited in their methods. The main reason comes from their training scheme that makes G *sample a single discrete item index*, which is quite different from G in the original GAN that generates *a vector* (e.g., images), each element of which has *continuous values*. Since the item indices are discrete and finite, it is highly probable for G to sample data that are *exactly* the same as those in the ground truth (i.e., a user's real choices). Hence, as the training progresses, D is thrown into confusion and thus gets degraded since it is trained with a large proportion of *contradicting labels* for the same item index: the same item can be sometimes labeled as "fake" when it is from G and other times as "real" when it is from the ground truth. This results in D sending a wrong signal to G, subsequently causing G to lower its likelihood to sample the ground truth items. As a consequence, instead of improving each other's capability, both G and D's accuracy will be worsened. This phenomenon will be shown clearly through our preliminary experiments in Section 3: after several epochs, the recommendation accuracy by D does not increase anymore and rather plummets, mainly due to the large portion of contradictorily labeled items. This results in preventing G from improving its accuracy in CF, where the competition between G and D does not help each other.

To address this problem, we propose a novel GAN based CF framework, named CFGAN. In our framework, given a user, G tries to generate her *plausible purchase vector* composed of real-valued elements, rather than to sample a single item index that she may be interested in and thus will purchase. Likewise, D tries to differentiate between this generated purchase vector of hers and her real purchase vector obtained from the ground truth. This approach to generating and discriminating real-valued vectors prevents D's confusion occurring in GraphGAN and IRGAN due to the items having contradictory labels since it is almost impossible for G to generate real-valued vectors that are completely identical to the vectors belonging to the ground truth. This property makes D guide G consistently to improve through back-propagation so that G's generated vectors get closer to the real purchase vectors as iterations proceed [45]. This novel CF framework would lead to successful adversarial training, thus achieving high accuracy in recommendation.

When transitioning the goal from discrete item index generation to real-valued vector generation for GAN-based CF, we identify a unique challenge that naturally emerges due to the characteristics of implicit feedback data. Unlike the image data considered by the original GAN [11, 32], which is represented as a *dense vector* composed of pixels with multiple bits depth, the ground truth data in CF is a *sparse single-valued vector* where an element is 1 if the corresponding user-item interaction is observed (i.e., a user purchases an item) and empty otherwise. While G tries to generate purchase vectors resembling that of the ground truth, it would be easily led astray, finding a useless but easy solution: i.e., generating the purchase vector having all of its elements as 1 without considering a user's relative preference at all.

Regarding this challenge, we propose three CF methods (i.e., *CFGAN_ZR*, *CFGAN_PM*, and *CFGAN_ZP*) developed on our CFGAN framework, each of which successfully addresses the challenge in its own way. Their idea in common is as follows: (1) to sample some unobserved elements; (2) to presume that they would correspond to negative feedback (i.e., their values are made not missing but zeros in the purchase vector); (3) to ask G to generate outputs close to 0s on the sampled elements and to 1s on the elements corresponding to a user's real purchases. Towards this goal, *CFGAN_ZR* uses *zero-reconstruction regularization* and *CFGAN_PM* takes *partial-masking* into account; *CFGAN_ZP* is a hybrid that uses both the zero-reconstruction regularization and the partial-masking. In addition, we formulate each of these methods by following *user-oriented* and *item-oriented* approaches.

We conduct extensive experiments using four real-world datasets with various evaluation metrics of precision, recall, nDCG, and MRR. The experimental results demonstrate not only the effectiveness of our *real-valued vector-wise adversarial training* in CF but also the superiority of our CFGAN-based CF methods, achieving significant improvements in terms of accuracy over state-of-the-art top-N recommenders such as BPR [34], FISM [18], and CDAE [44].

In summary, the main contributions of this paper are listed as follows:

- We identify the limitation of existing GAN-based CF methods and also highlight it through our preliminary experiments.
- We suggest a new direction, real-valued vector-wise adversarial training, that GAN-based CF methods should follow in order to fully exploit the advantage of adversarial training for higher recommendation accuracy in CF.
- We propose a generic GAN-based CF framework, named CFGAN, that follows our suggested direction of real-valued vector-wise adversarial training.
- We propose three CF methods realized under CFGAN framework, named *CFGAN_ZR*, *CFGAN_PM*, and *CFGAN_ZP*,

which successfully remedy the challenge arising when vector-wise adversarial training is employed for CF.
- We validate the effectiveness of the suggested direction to vector-wise adversarial training, and also demonstrate the superiority of our CF methods over state-of-the-art top-N recommenders.

2 PRELIMINARIES

2.1 Model-based CF

Let $U = \{u_1, u_2, .., u_m\}$ and $I = \{i_1, i_2..., i_n\}$ denote a set of m users and a set of n items, respectively. We define I_u as a set of items receiving implicit feedback from (i.e., purchased by) u. Users' implicit feedback on items is represented by a sparse matrix $\mathbb{R} = (r_{ui})_{m \times n}$, where an element r_{ui} is 1 if i is contained in I_u and empty (i.e., unobserved) otherwise. We denote the purchase vectors of user u and item i as $\mathbf{r_u}$ and $\mathbf{r_i}$, respectively. The goal of CF is to predict u's preference score on j (denoted as $\hat{r}_{uj}$ where $j \in I \setminus I_u$), reflecting how much likely it is for u to purchase i, and then recommend a subset of items whose predicted scores are the highest to u. From the viewpoint of model-based ones, CF computes $\hat{r}_{uj}$ by using a model, which can be formally denoted as $\hat{r}_{uj} = f(u, i|\Theta)$ where f denotes the model, i.e., a function that is parameterized by Θ and maps a given pair of user u and item i to a preference score [13]. The set of model parameters Θ can be learned by optimizing an objective function with $\mathbb{R}$ as training data.

By and large, existing model-based CF methods could be classified according to the following two types of an objective function used to train the model: *pointwise* methods and *pairwise* methods. The former aims to minimize the mean squared error between $\hat{r}_{uj}$ and its original value r_{ui}, while the latter maximizes the difference of the predicted preferences between more preferred (i.e., purchased) and less preferred (i.e., non-purchased) items.

2.2 GAN-based CF

Recently, Goodfellow et al. proposed Generative Adversarial Network (GAN) [11], which provides a new way to learn machine learning models. Through a competition process involving a generative model (shortly, G) and a discriminative model (shortly, D), G learns to capture the distribution of ground-truth data so that it can generate plausible synthetic data whose characteristics are not different from those of ground-truth. Formally, G and D are playing the following two-player minimax game with the objective function V:

$$\max_{\theta} \min_{\phi} V(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\phi}}[\log(1 - D(\hat{\mathbf{x}}))] \quad (1)$$

where ϕ and θ denote the model parameters of G and D, respectively. $\mathbf{x}$ is a ground-truth data from the data distribution p_{data} while $\hat{\mathbf{x}}$ is a synthetic data from the model distribution p_{ϕ}, typically derived by $\hat{\mathbf{x}} = G(\mathbf{z})$ where $\mathbf{z}$ is a random noise vector [11, 12]. $D(\cdot)$ indicates the estimated probability of its input being a ground-truth. In general, both G and D are neural networks and iterate in optimizing the respective model parameters while minimizing and maximizing the same objective function, respectively [5].

Although the most successful domain was image generation, GAN also has great potential to achieve more promising results in other domains [23], e.g., WaveGAN [8] in music generation and SeqGAN [45] in sentence generation. In the recommender systems area, IRGAN [42] and GraphGAN [41] are the pioneering methods that successfully apply GAN to CF[1]. They try to achieve more satisfactory accuracy in recommendation by borrowing the adversarial training approach, rather than optimizing the traditional pointwise or pairwise objective functions to train the CF model. Despite the fact that IRGAN and GraphGAN have been independently developed, they are quite similar to each other in their philosophy used in regards to CF. Hence, we describe the philosophy, based on IRGAN.

In IRGAN, G tries to generate (or, sample) the indices of items relevant to a given user, and D tries to discriminate the user's ground truth items from those synthetically generated by G. More specifically, G is implemented as a softmax function, denoted as:

$$P_{\phi}(i_k|u) = \frac{\exp(s_{\phi}(u, i_k))}{\sum_i \exp(s_{\phi}(u, i))} \quad (2)$$

where $P_{\phi}(i_k|u)$ is G's conditional probability distribution of sampling the item at index k given u, and $s_{\phi}(u, i_k)$ is a scoring function that reflects the chance of i_k being purchased by u. On the other hand, D estimates the probability of each item i belonging to the ground truth of given user u by using a sigmoid function of the scoring function $s_{\theta}(u, i)$:

$$D(i|u) = \sigma(s_{\theta}(u, i)) = \frac{\exp(s_{\theta}(u, i))}{1 + \exp(s_{\theta}(u, i))} \quad (3)$$

where $s_{\theta}(u, i)$ reflects the relevance of i to u. Both the scoring functions s_{ϕ} and s_{θ} are based on matrix factorization models, formally, $s_{\theta}(u, i) = \mathbf{d_u^T d_i} + d_i^b$ and $s_{\phi}(u, i) = \mathbf{g_u^T g_i} + g_i^b$, where $\mathbf{d_u}$ and $\mathbf{d_i}$ are the latent factors belonging to D, respectively; $\mathbf{g_u}$ and $\mathbf{g_i}$ are the latent factors belonging to G; d_i^b and g_i^b are their item bias terms. Finally, IRGAN's objective function $V(G, D)$ is formulated as:

$$V(G, D) = \Sigma_u \big(\mathbb{E}_{i \sim P_{data}(i|u)}[\log D(i|u)] + \mathbb{E}_{\hat{i} \sim P_{\phi}(\hat{i}|u)}[\log(1 - D(\hat{i}|u))]\big) \quad (4)$$

While D can be solved by stochastic gradient ascent, G cannot receive gradient from D since sampling of i is discrete. Hence, IRGAN employs the policy-gradient based reinforcement learning [43] to update G.

Although they are not necessarily pure CF methods, there have also been other research results on recommender systems based on GAN, especially focusing on visual aspects of items, e.g., [19], [37], and [16]. They go beyond the scope of our research as they all require additional visual information of items.

3 MOTIVATION

Despite the fact that IRGAN and GraphGAN opened up new possibilities for GAN to be successful in CF, there are still rooms for further improvement. In this section, we point out the limitation in their "discrete item index generation" approach, which made us motivated to improve the accuracy in CF by overcoming this limitation.

Figure 1 briefly illustrates their training process in common. In the initial stage (Figure 1(a)), G (almost) randomly samples items but improves (Figure 1(b)) its performance with the guidance of D. As the training is further progressed (Figure 1(c)), G gets to

[1]These methods are not the models dedicated for CF. In addition to recommendation tasks, IRGAN can be applied to web search and question answering, and GraphGAN can be applied to link prediction and node classification.

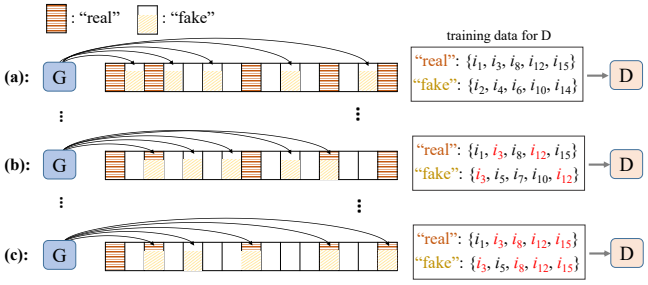


Figure 1: Discrete item index generation.

sample frequently the items exactly same as those in ground truth. Evidently, from D's perspective, this is *much more than a challenge* since the same item index (e.g., i_3 in Figure 1) is sometimes labeled as "fake" (i.e., from G) and other times as "real" (i.e., from the ground truth), which makes D significantly confused, thereby leading to its degradation. This may subsequently make D give to G a wrong signal, through the policy gradient, that G should lower its probability to sample those items in the ground truth, which may result in G's degradation as well.

To highlight the aforementioned limitation quantitatively, we conduct preliminary experiments on real-world datasets. Here, we focus on presenting the experimental results, skipping the details of the experimental environment including datasets and accuracy measures, which can be found in Section 5.1. In Figures 2 and 3, we visualize the learning curves of GraphGAN and IRGAN, where each graph reports the accuracy of top-5 recommendations. Figure 4 shows the proportion of contradictory labeled items among those generated by G as the adversarial training progresses. As shown in Figures 2 and 3, the training seems to work well in a few initial epochs: G pushes D to its limit by sampling challenging items, and D guides G. However, after these epochs, the recommendation accuracy by D does not increase anymore rather suddenly plummets. This is mainly because there are a large proportion of contradictory labeled items in the training data (see Figure 4). Eventually, G is not able to perform better anymore and even gets to perform worse since the reward from D could not give feedback beneficial to G anymore. As a result, in this situation, the competition process between G and D could not produce the synergy effect in providing a high-quality recommendation list.

Now our question is, how could we fully enjoy the advantage of the adversarial training on the task of item recommendation? More specifically, how could we train G and D properly so that G can consistently push D to its limit, without pushing it "well beyond" its limit, and D can consistently give only the feedback beneficial to G? To answer this question, we suggest a new direction of "vector-wise training", where G generates *real-valued vectors* and D differentiates between those from G and the real vectors coming from the ground truth. More specifically, given a user, G tries to generate her purchase vector that seems plausible while D tries to determine correctly whether a given purchase vector is her real one or the fake one generated by G. Under this vector-wise training, G may be able to generate the purchase vector that is very close (but not identical) to the real one, while it has little possibility to generate a purchase vector exactly same as the real one. This makes D rarely fooled by G, thereby enabling D to break its own limit consistently over a series of epochs owing to the qualified training data from G. Likewise, D can provide a beneficial training signal consistently to G that shows how G should slightly change its outputs to produce more-realistic fake purchase vectors through back-propagation [45]. We believe that our proposed vector-wise training could make adversarial training successful by fully exploiting the advantage of GAN, which in turn produces high-quality recommendations.

4 CFGAN

We start by introducing a generic CF framework based on GAN, called CFGAN, which follows the suggested direction of vector-wise training (Section 4.1). Then, we highlight a challenge that our CFGAN faces and propose a suite of CF methods that have been realized on top of our CFGAN framework while addressing the challenge successfully (Section 4.2). Note that these CF methods could be realized in two different fashions: user-oriented and item-oriented. For simplicity, we basically present our CFGAN in the perspective of a user-oriented fashion, which regards each user as a training instance, up to Section 4.2.3. Then, we discuss how to realize this framework in an item-oriented version in Section 4.2.4.

4.1 Proposed framework

Figure 5 shows the overview of our CFGAN framework. As in the other GAN-based CF methods, CFGAN is also based on the conditional GAN framework [28], where both G and D are user-conditional, implying that the model parameters are learned while taking each user's personalization into account. Given a user-specific condition vector $\mathbf{c_u}$ and a random noise vector $\mathbf{z}$[2], G in our CFGAN generates an n-dimensional purchase vector $\hat{\mathbf{r}}_\mathbf{u}$, which is expected to be a sparse vector where all the elements corresponding to u's purchased items, $\hat{r}_{ui}$ where $i \in I_u$, are hopefully 1. Likewise, conditioned by $\mathbf{c_u}$, D is trained to distinguish the generated purchase vector from u's real one. Formally, our D's objective function, denoted as J^D, is as follows[3]:

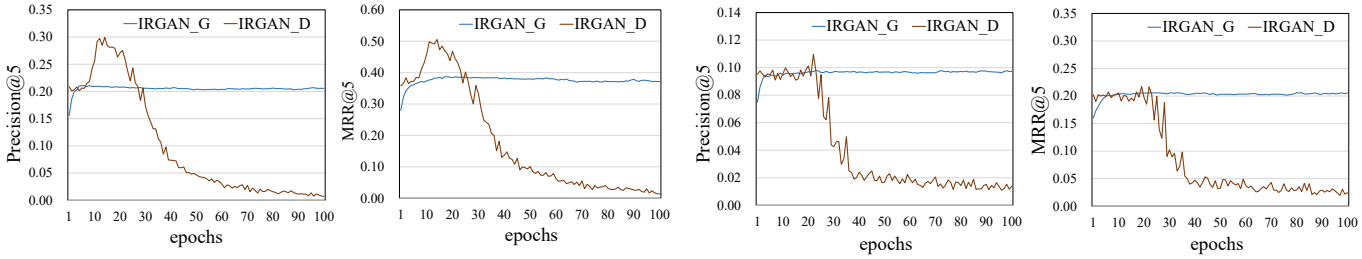
$$\begin{aligned} J^D &= -\mathbb{E}_{\mathbf{x}\sim P_{data}}[\log D(\mathbf{x}|c)] - \mathbb{E}_{\hat{\mathbf{x}}\sim P_\phi}[\log(1 - D(\hat{\mathbf{x}}|c)] \\ &= -\Sigma_u \log D(\mathbf{r_u}|\mathbf{c_u}) - \Sigma_u \log(1 - D((\hat{\mathbf{r}}_\mathbf{u} \odot \mathbf{e_u})|\mathbf{c_u})) \\ &= -\Sigma_u \big(\log D(\mathbf{r_u}|\mathbf{c_u}) + \log(1 - D((\hat{\mathbf{r}}_\mathbf{u} \odot \mathbf{e_u})|\mathbf{c_u}))\big) \end{aligned} \quad (5)$$

And, similarly, that of our G is[4]:

$$J^G = \Sigma_u \log \big(1 - D((\hat{\mathbf{r}}_\mathbf{u} \odot \mathbf{e_u})|\mathbf{c_u})\big) \quad (6)$$

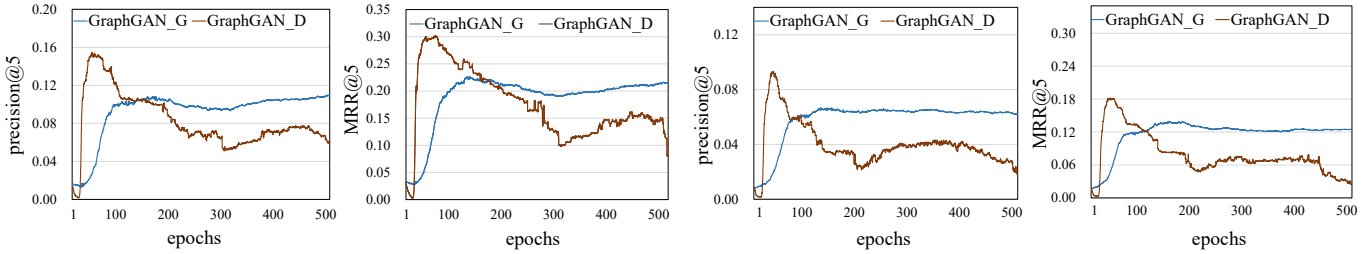
where $\mathbf{e_u}$ in Eqs. (5) and (6) is an n-dimensional indicator vector specifying whether u has purchased each item i (e_{ui} =1) or not (e_{ui}=0), and $\odot$ stands for element-wise multiplication. The main difference between our CFGAN and the original GAN is that we mask G's output $\hat{\mathbf{r}}_\mathbf{u}$ by multiplying it with $\mathbf{e_u}$, in order to deal with the sparsity in ground-truth data that G tries to mimic. By doing so, only G's output on the purchased items can contribute to the learning of G and D. In other words, we "drop" G's output nodes corresponding to the non-purchased items, so that (1) D disregards the dropped nodes and (2) G does not get the gradient of

[2]The implementation details of $\mathbf{c_u}$ and $\mathbf{z}$ will be provided in Section 5.1.3.
[3]For simplicity, we define both of the objective functions to be a minimization problem.
[4]In practice, however, we choose to minimize $-\log D(\cdot)$ rather than $\log(1 - D(\cdot))$, which has been known to provide stronger gradients in the early stage of the training [5, 11].



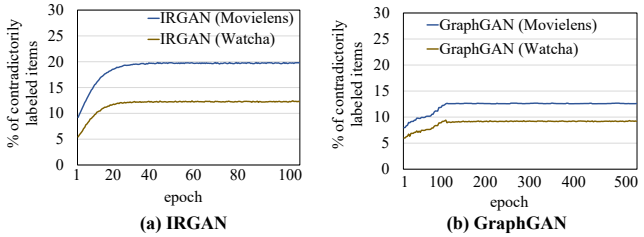
(a) Results on Movielens 100K (b) Results on Watcha

Figure 2: Learning trend of IRGAN.



(a) Results on Movielens 100K (b) Results on Watcha

Figure 3: Learning trend of GraphGAN.



(a) IRGAN (b) GraphGAN

Figure 4: The proportion of contradictory labeled items among G's generations.

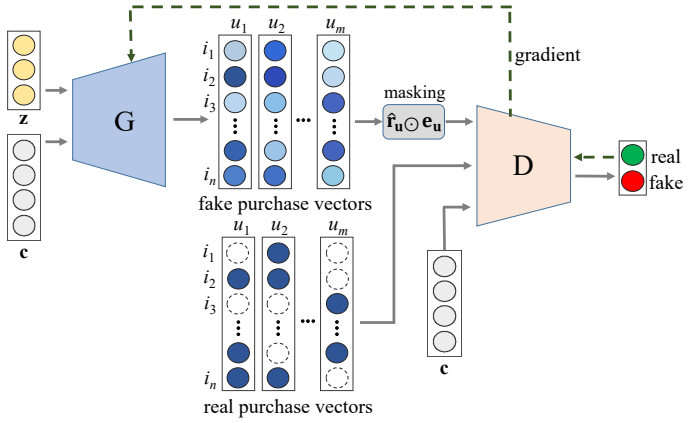


Figure 5: Overview of our CFGAN framework.

the loss from D w.r.t. the dropped nodes. This idea has the common spirit shared with other CFs using pointwise objective functions in that they build a model based on the observed user-item purchase records that can be used later to identify potential purchase [31].

We implement both G and D as multi-layer neural networks, parameterized by ϕ and θ, respectively. Specifically, G is an L^G-layer (L^G>=2) neural network with input $\{\mathbf{z}, \mathbf{c_u}\}$ that outputs $\hat{\mathbf{r}}_\mathbf{u}$, where $\{\}$ indicates the concatenation of two vectors inside. D is an L^D-layer (L^D>=2) neural network with input whether $\hat{\mathbf{r}}_\mathbf{u} \odot \mathbf{e_u}$ (i.e., from G) or $\mathbf{r_u}$ (i.e., from the ground-truth) concatenated with $\mathbf{c_u}$, and outputs a single scalar value representing the probability that its input came from the ground truth, rather than G. We employ the stochastic gradient descent with minibatch and back-propagation to train our G and D. We update their respective parameters ϕ and θ alternately, keeping one fixed when the other is being updated.

After the adversarial training is completed, fed with $\mathbf{z}$ and $\mathbf{c_u}$, G generates a dense vector $\hat{\mathbf{r}}_\mathbf{u}$, which contains predicted preference scores for all items in the dataset. We use the j-th element in $\hat{\mathbf{r}}_\mathbf{u}$ as the predicted preference score of user u on each item j where $j \in I \setminus I_u$. Finally, the top-N items whose predicted scores are the highest would be picked and recommended to u.

4.2 CF methods

While following our suggested direction of vector-wise adversarial training, CFGAN faces a unique challenge that was not encountered by the original GAN, which aimed at image generation. Unlike the image data, which is typically represented as a vector composed of pixels with multiple bits depth, the implicit feedback data for a user in recommendation is a very sparse, single-valued vector where an element is 1 if the corresponding user-item interaction (i.e., purchase) is observed and empty otherwise. While G tries to deceive D by generating purchase vectors resembling that of the ground truth, it would lead to a trivial solution that simply predicts all the outputs in the purchase vector as 1. This solution is easy to

obtain but is useless because it does not capture a user's *relative preference* at all, which is fundamental for CF.

This section proposes a suite of novel CF methods, which are developed on top of our proposed CFGAN framework, to address the above challenge. The core idea shared by them are as follows: In every training iteration (say, t), we randomly choose a portion of each user's non-purchased items ($I \setminus I_u$), and presume them as *negative items*, indicating their corresponding feedback is *not missing but zero*. Then, we train G to generate a user's purchase vector in such a way that the value on her negative items is close to zero. By doing so, we expect that G would not only focus on generating the value to be close to 1 on the purchased items in order to deceive D, but also consider producing low values on the negative items. As a result, it is avoided to accept G's trivial solution that simply outputs 1 for all the elements in the purchase vector.

Formally, let $N_u^{(t)}$ denote a set of negative items sampled on the t-th iteration. The corresponding entries in purchase matrix $\mathbb{R}$ are regarded as not missing but zero only in this iteration. We denote S to be a portion of items (in percentage) identified from $I \setminus I_u$. For example, if S is set as 30, then 30 percent of items in $I \setminus I_u$ will be selected. Depending on how the identified negative items are used, we propose three CF methods: the first one is based on zero-reconstruction regularization (named *CFGAN_ZR*), the second one is based on partial-masking (named *CFGAN_PM*), and the last one is the hybrid of the two (named *CFGAN_ZP*).

4.2.1 CFGAN_ZR. In this method, we denote $N_u^{ZR(t)}$ and S^{ZR} to indicate the selected negative items and the parameter S specifying the portion of negative items selected from non-purchased items (in percentage), respectively. In addition to our G's objective function defined in Eq. (6), we use a reconstruction loss function as a regularization term as follows:

$$J^G = \Sigma_u \big(\log(1 - D((\hat{\mathbf{r}}_\mathbf{u} \odot \mathbf{e}_\mathbf{u})|\mathbf{c}_\mathbf{u})) + \alpha \cdot \Sigma_j (x_{uj} - \hat{x}_{uj})^2 \big) \quad (7)$$

Here, the second term in $\Sigma_u(\cdot)$ corresponds to the zero-reconstruction (in short, ZR) regularization where $j \in N_u^{ZR(t)}$ and α is a tunable parameter for controlling the importance of the ZR term in the entire equation. By minimizing J^G, G would focus on generating scores close to 1 on the observed entries and also producing low values (i.e., close to zero) on the unobserved entries, thus preventing G from reaching a trivial and useless solution.

4.2.2 CFGAN_PM. Here, we use the symbols $N_u^{PM(t)}$ and S^{PM} to denote selected negative items and the sampling parameter S, respectively. In this method, we do not necessarily drop all the output nodes corresponding to non-purchased items, but let those corresponding to $N_u^{PM(t)}$ remain. Thereby, D now exploits the input values from not only the purchased items but also the negative items in its discrimination task. Moreover, the gradient of the loss from D w.r.t. the output on the selected negative items can be passed back to G. Consequently, G is guided to make its generated output closer to 1 for the purchased items, and closer to 0 for the selected negative items.

The aforementioned idea is applied to J^G and J^D as follows:

$$J^D = -\Sigma_u \big(\log D(\mathbf{r}_\mathbf{u}|\mathbf{c}_\mathbf{u}) + \log(1 - D(\hat{\mathbf{r}}_\mathbf{u} \odot (\mathbf{e}_\mathbf{u} + \mathbf{k}_\mathbf{u})|\mathbf{c}_\mathbf{u})) \big) \quad (8)$$

$$J^G = \Sigma_u \log \big(1 - D(\hat{\mathbf{r}}_\mathbf{u} \odot (\mathbf{e}_\mathbf{u} + \mathbf{k}_\mathbf{u})|\mathbf{c}_\mathbf{u})\big) \quad (9)$$

Algorithm 1 *CFGAN_ZP*

Input: implicit feedback matrix $\mathbb{R}$, learning rate for G and D μ^G and μ^D, minibatch size for G and D M^G and M^D
Output: G's parameters ϕ

1: Initialize θ and ϕ
2: **while** not converged **do**
3: **for each** $u \in U$ **do**
4: Sample $N_u^{ZR(t)}$
5: Sample $N_u^{PM(t)}$
6: **end for**
7: **for** G-step **do**
8: Sample minibatch of M^G users
9: Generate fake purchase vectors $\{\hat{\mathbf{r}}_\mathbf{1}, \hat{\mathbf{r}}_\mathbf{2}, \ldots, \hat{\mathbf{r}}_{\mathbf{M^G}}\} \sim G$
10: Update G by $\phi \leftarrow \phi - \frac{\mu^G}{M^G} \cdot \nabla_\phi J^G$
11: **end for**
12: **for** D-step **do**
13: Sample minibatch of M^D users
14: Get their real purchase vectors $\{\mathbf{r}_\mathbf{1}, \mathbf{r}_\mathbf{2}, \ldots, \mathbf{r}_{\mathbf{M^D}}\}$
15: Generate fake purchase vectors $\{\hat{\mathbf{r}}_\mathbf{1}, \hat{\mathbf{r}}_\mathbf{2}, \ldots, \hat{\mathbf{r}}_{\mathbf{M^D}}\} \sim G$
16: Update D by $\theta \leftarrow \theta - \frac{\mu^D}{M^D} \cdot \nabla_\theta J^D$
17: **end for**
18: **end while**
19: **return** ϕ

where $\mathbf{k}_\mathbf{u}$ is an n-dimensional indicator vector such that $k_{uj} = 1$ if $j \in N_u^{PM(t)}$ and $k_{uj} = 0$ otherwise.

4.2.3 CFGAN_ZP. This is a hybrid method that uses both the zero-reconstruction regularization and the partial-masking, so that the advantages of both methods can be exploited. The sets of selected negative items, $N_u^{ZR(t)}$ and $N_u^{PM(t)}$, are used for the zero-reconstruction and the partial-masking, respectively. Here, the objective function of D is the same as that in Eq. (8). G's objective function is denoted as:

$$J^G = \Sigma_u \big(\log(1 - D(\hat{\mathbf{r}}_\mathbf{u} \odot (\mathbf{e}_\mathbf{u} + \mathbf{k}_\mathbf{u})|\mathbf{c}_\mathbf{u})) + \alpha \cdot \Sigma_j (x_{uj} - \hat{x}_{uj})^2 \big) \quad (10)$$

Due to space limitations, we provide here only the learning algorithm for *CFGAN_ZP*, as shown in Algorithm 1. *CFGAN_ZR* and *CFGAN_PM* can be regarded as the special cases of *CFGAN_ZP*: *CFGAN_ZR* is *CFGAN_ZP* that skips line 5 in Algorithm 1 while *CFGAN_PM* is *CFGAN_ZP* that skips line 4.

4.2.4 CFGAN in item-oriented fashion. Until now, we have described our CFGAN in a user-oriented fashion, where each training instance corresponds to a user, and G (and D) generates (and discriminates) a user's purchase vector. Our framework, however, could be realized as well in an item-oriented fashion, where a training instance corresponds to an item, rather than a user. Specifically, conditioned by item-specific information $\mathbf{c_i}$, G tries to generate each item's plausible purchase vector, which is an m-dimensional vector where m is the number of users, indicating how likely it is for a user corresponding to each element in the vector to purchase the item. Likewise, conditioned by item-specific information, D tries to discriminate the item's ground truth purchase vector from that

generated by G. After training, G predicts the preference scores for all users given each item i. Finally, the N items with the highest predicted scores are suggested to each user.

Each of our proposed CF methods, *CFGAN_ZR*, *CFGAN_PM*, and *CFGAN_ZP*, can be realized with both user-oriented and item-oriented formulations. Thus, all the proposed CF methods can now be denoted as follows: *uCFGAN_ZR*, *uCFGAN_PM*, and *uCFGAN_ZP*, if they are user-oriented ones; *iCFGAN_ZR*, *iCFGAN_PM*, and *iCFGAN_ZP*, if they are item-oriented ones.

5 EVALUATION

This section reports and analyzes the results of our extensive experiments to validate the effectiveness of our proposed ideas. Our experiments are designed to answer the following three key questions:

- Q1. How much effective is our proposed vector-wise adversarial training for the CF task?
- Q2. How much does the accuracy of our proposed CF methods vary with different values of key hyper-parameters?
- Q3. How much accurate are the proposed methods compared with the state-of-the-art top-N recommenders?

5.1 Experimental settings

5.1.1 Dataset. We used four real-world datasets: Watcha[5], Ciao [39], Movielens 100K, and Movielens 1M datasets. Table 1 summarizes their detailed statistics. Even if all those datasets provide users' explicit ratings on items, we convert them into 1 (i.e., indicating a purchase, or any other implicit feedback), which has been popularly done in other CF researches to get implicit feedback data [15, 34, 44]. For each dataset, we randomly split its user-item interactions into two subsets: 80% for training and the rest 20% for testing. Then, we again set aside 20% of the training data for validation in order to use it for tuning hyper-parameters.

Table 1: Dataset statistics

Datasets	# users	# items	# purchases	Sparsity
Ciao	996	1,927	18,648	98.72%
Watcha	1,391	1,927	101,073	96.98%
Movielens 100K	943	1,682	100,000	93.69%
Movielens 1M	6,039	3,883	1,000,209	95.72%

5.1.2 Evaluation metrics. We employ four popular accuracy metrics for top-N recommendations: precision (P@N), recall (R@N), normalized discounted cumulative gain (G@N), and mean reciprocal rank (M@N). The first two metrics, P@N and R@N, focus on how many correct items are included in the recommendation list while the latter two metrics, G@N and M@N, account for the ranked position of correct items in the recommendation list. We set N as 5 and 20.

5.1.3 Implementation details. We first determine some hyper-parameters that empirically perform well regardless of the combination of the other hyper-parameters or datasets used: we use a sigmoid function as an activation function for neural networks; for initializing their edge weights, we use the popular Xavier's approach [10]. We perform grid search to tune the other hyper-parameters by evaluating the accuracy on the held-out validation set. We varied the number of hidden layers of G and D with $\{1, 2, 3, 4, 5\}$, the number of hidden nodes per hidden layer with $\{50, 100, 150, 200, 300\}$, a learning rate with $\{0.005, 0.001, 0.0005, 0.0001\}$, a size of a mini-batch as $\{32, 64, 128, 256\}$, each of S^{ZR} and S^{PM} with $\{10, 30, 50, 70, 90\}$, and ZR-coefficient α with $\{0.5, 0.25, 0.1, 0.05, 0.01\}$.

The user-specific condition $\mathbf{c_u}$ could be any kind of information that characterizes a user, such as gender, age, and previous purchases. (resp. Likewise, the item-specific condition $\mathbf{c_i}$ could be an item's images, tags, and descriptions. In our methods, we use a user's (or, an item's) purchase vector to specify a user (or, an item). In addition, as in the other GAN-based CFs [41, 42], we do not use the random noise variable $\mathbf{z}$ since our goal is to generate a single, most plausible recommendation result to a target user rather than multiple outcomes.

5.2 Q1: Effectiveness of vector-wise adversarial training

One of our contributions is to highlight the limitation of the discrete item index generation that existing GAN-based CFs follow, and to suggest a new direction of real-valued, vector-wise adversarial training. Note that we already showed quantitatively the problem of IRGAN and GraphGAN in Section 3. The experiments here focus on showing the effectiveness of our vector-wise training. To this end, we train our proposed CF methods and visualize the curves of their prediction accuracy obtained in each training epoch. In addition, although our methods do not sample the item index at all, we let G in our methods sample items for the purpose of estimating the proportion of ground truth items among the total items generated by G. As in IRGAN, we use softmax function on the predicted preferences of each user on items, as $\frac{\exp(\hat{r}_{ui_k})}{\sum_{i \in I} \exp(\hat{r}_{ui})}$.

Figures 6 and 7 report the learning curves provided by each method, and Figure 8 shows the portion of ground truth items among the items generated by G. Due to space limitations, we only show the results of the accuracy in top-5 recommendation on Movielens and Watcha datasets. Those of top-20 on the other datasets exhibit very similar tendency.

Compared to IRGAN and GraphGAN shown in Figures 2 and 3, our methods seem to consistently exploit the benefits of adversarial training throughout the entire process. Hence, the accuracy was improved up to 72.6% compared to IRGAN, and up to 104.3% compared to GraphGAN, in terms of P@5. This is mainly because we follow the vector-wise adversarial training, where competition between G and D consistently encourages both of them to improve their capability, without D's confusion due to the existence of contradictorily labeled data. This property makes G consistently push D to its limit by generating a purchase vector that seems more and more plausible; D also consistently gives G beneficial feedback that tells how G should change its outputs to produce more realistic purchase vectors. By fully enjoying the advantage of adversarial training, all our methods will eventually be able to produce high-quality recommendations.

In Figure 8, we see that our proposed methods sample more and more ground truth items assuming G in our methods also performs discrete item sampling. If our method followed the discrete

[5]Watcha is a privately released dataset from a Korean movie recommendation company (http://watcha.net)

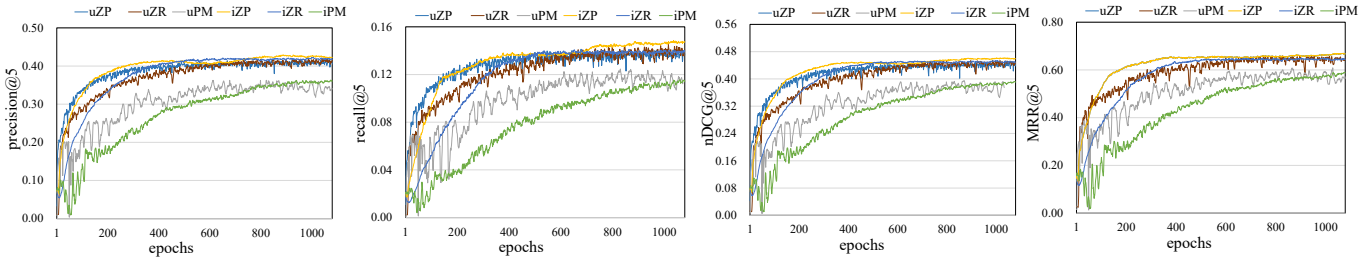


Figure 6: Learning curves of our methods on Movielens 100K.

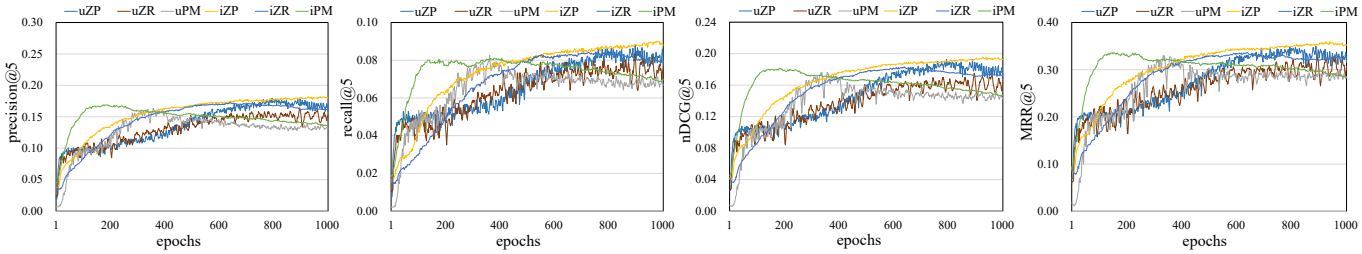


Figure 7: Learning curves of our methods on Watcha.

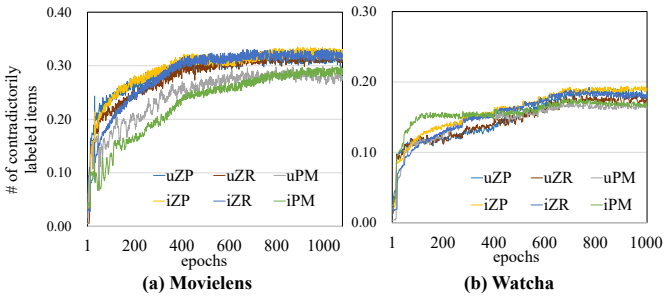


Figure 8: The proportion of contradictory labeled items among G's generations.

item generation approach, the performance of G would not have improved beyond a certain level due to D's confusion. However, by following the proposed vector-wise adversarial training, D is rarely fooled by G and breaks its own limit continuously over epochs owing to the qualified training data from G, which subsequently results in G's accuracy improvement.

5.3 Q2: Influence of key hyper-parameters

All the hyper-parameters used in our methods are shown in Section 5.1.3, each of which affects the recommendation accuracy. Among them, this subsection investigates the impact of the following hyper-parameters unique in our proposed methods: the sampling parameters S^{ZR} and S^{PM}, and the zero-reconstruction regularization coefficient α. The results on only Movielens 100K are shown due to limited space.

Figure 9 reports the accuracy of our methods with regards to S^{ZR} and S^{PM}. Here, we fix α as 0.1. We observe that when S^{PM} and S^{ZR} are reasonably high (i.e., more than 50), the proposed methods have the highest accuracy. For this reason, in the following experiments, we set the values of S^{PM} and S^{ZR} as 70. Next, Figure 10 shows the performance of our methods depending on the value of α. We observe that the moderate range of α values would be in [0.05, 0.2]. If it is set as a smaller value (e.g., less than 0.01), G is less affected by the zero-reconstruction regularization term so that it is difficult for G to avoid the trouble of producing a trivial solution. In contrast, if it is set as a larger value (e.g., more than 0.5), G would pay too much attention to making outputs close to zero, rather than achieving its original goal of generating plausible purchase vectors. Both aforementioned cases would lower the accuracy in recommendation. In the following experiments, we set α as 0.1, which we believe is a good balance between GAN loss and zero-reconstruction regularization.

Furthermore, from the above experimental results, we can see that *iCFGAN_ZP* consistently outperforms the other proposed CF methods. Thus, in the next set of experiments, we fix our method as *iCFGAN_ZP* and compare it with the state-of-the-arts.

5.4 Q3: Comparisons with the state-of-the-arts

We now compare the accuracy of the proposed methods with those of the state-of-the-art top-N recommenders, which are listed as follows:

- **ItemPop**: It is the simplest non-personalized algorithm that ranks items by the descending order of popularity (i.e., the number of purchase records).
- **BPR** [34]: It optimizes the relative order of the preferences for purchased and non-purchased item pairs. We also employ the adaptive sampling strategy proposed in [33] to boost convergence as well as accuracy.
- **FISM** [18]: It learns the item-item similarity matrix as a product of two low-dimensional latent factor matrices.
- **CDAE** [44]: It uses a denoising autoencoder structure for CF while integrating user-specific latent features. It is an important baseline since CDAE is trained with a user's purchase vector as input and aims at reconstructing it as similarly as

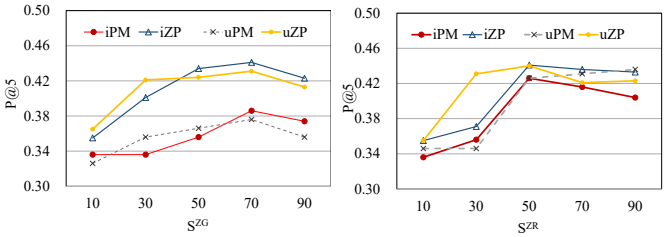


Figure 9: Accuracy depending on S^{PM} and S^{ZR}.

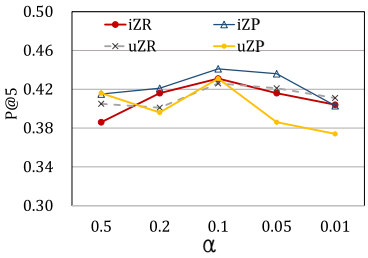


Figure 10: Accuracy depending on α.

possible, just like our model. However, it uses the traditional pointwise objective function to learn model while we use the conditional GAN framework.

- **IRGAN** [42] and **GraphGAN** [41]: IRGAN is described in detail in Section 2.2. GraphGAN uses *graph softmax*, which was proposed so that the softmax function can consider the graph structure (e.g., user-item bipartite graph).

For each model, we test various values for its hyper-parameters based on the values mentioned in the papers (e.g., the number of latent factors in MF-based models, and the number of hidden layers and nodes in CDAE) and select a set of values that provide the highest accuracy on the held-out validation set.

Tables 2 to 3 report the comparison results for all the recommenders with the four datasets. As shown in the tables, our methods perform better than CDAE, which is most similar to our CF methods in terms of basic philosophy, on all the datasets in terms of all the four metrics. We believe that this accuracy improvement mainly comes from the difference in the training approaches (i.e., pointwise optimization and adversarial training) used by ours and CDAE. By successfully applying adversarial training to CF, we succeeded in recovering a more-plausible purchase vector than CDAE that employs the traditional pointwise objective function.

Overall, our proposed methods universally and consistently provide the best accuracy. Notably, *iCFGAN_ZP* achieves 21.3% relative improvement over the best competitor (i.e., CDAE in this case) in terms of M@5 on Ciao. It also achieves 2.8% improvement compared to our best competitor (i.e., FISM in this case) in regards to P@5 on Movielens 1M. We believe that the benefits are once again credited to our vector-wise adversarial training approach, where more plausible prediction scores are predicted as a result of G and D's competition.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we identified the limitation of existing GAN-based CF methods that generate and distinguish discrete item indices. In order to address the limitation, we suggested a new direction of *real-valued, vector-wise training* that takes full advantage of adversarial training in GAN. We proposed a novel GAN-based CF framework (CFGAN) based on our suggested direction. G in our CFGAN aims at generating a realistic purchase vector with real-valued elements, and D tries to distinguish between the vectors generated by G and the real ones from the ground truth. We proposed three CF methods realized under our CFGAN, which all address the unique challenge arising when vector-wise adversarial training is applied to CF: the first one named *CFGAN_ZR* is based on zero-reconstruction regularization, the second one called *CFGAN_PM* is based on partial-masking, and the last one *CFGAN_ZP* is the hybrid of the two. We conducted extensive experiments on four datasets to verify the effectiveness of vector-wise adversarial training as well as our CFGAN. Finally, we demonstrated our CF methods realized on CFGAN outperform significantly state-of-the-art top-N recommenders in terms of accuracy.

In future work, we plan to investigate some recent state-of-the-art GANs whose convergence properties and optimization stability are quite improved (e.g., WGAN [3], LSGAN [27], and BEGAN [4]), and apply their techniques to our CFGAN in order to further improve the model stability and recommendation accuracy. We also plan to extend CFGAN to a hybrid manner that involves possible item side information and user profiles by considering them as item-specific and user-specific conditions (i.e., $\mathbf{c_i}$ and $\mathbf{c_u}$), respectively. Moreover, motivated by so-called data augmentation performed by GAN [2, 9], we are particularly interested in utilizing GAN to generate plausible *ratings* by learning user-item rating dataset, in order to deal with the well-known data sparsity problem [17, 24, 26]. Lastly, we also plan to apply our CFGAN to other related problems such as link prediction.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT: Ministry of Science and ICT) (No. NRF-2017R1A2B3004581) and Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. NRF-2017M3C4A7083678). Also, we thank the Naver Corporation for their support including computing environment and data, which helped us greatly in performing this research successfully.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.
[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).
[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).
[4] David Berthelot, Thomas Schumm, and Luke Metz. 2017. BEGAN: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717* (2017).
[5] Edward Choi et al. 2017. Generating multi-label discrete electronic health records using generative adversarial networks. *arXiv preprint arXiv:1703.06490* (2017).

Table 2: Comparison results in Ciao and Watcha

datasets	Ciao								Watcha							
metrics	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20
ItemPop	.031	.024	.040	.127	.047	.065	.056	.067	.051	.044	.054	.151	.060	.102	.094	.126
BPR	.036	.025	.040	.141	.052	.066	.066	.078	.096	.077	.055	.166	.105	.127	.205	.238
FISM	.062	.040	.072	.178	.079	.109	.127	.147	.184	.132	.090	.248	.196	.221	.354	.387
CDAE	.061	.042	.075	.185	.081	.108	.127	.151	.179	.129	.089	.239	.193	.217	.353	.384
GraphGAN	.026	.017	.041	.100	.041	.058	.057	.068	.065	.042	.047	.130	.071	.102	.097	.123
IRGAN	.035	.023	.042	.111	.046	.066	.082	.088	.100	.078	.051	.148	.107	.122	.210	.225
Ours	**.072**	**.045**	**.081**	**.194**	**.092**	**.124**	**.154**	**.167**	**.188**	**.132**	**.091**	**.248**	**.200**	**.222**	**.360**	**.394**

Table 3: Comparison results in Movielens 100K and Movielens 1M

datasets	Movielens 100K								Movielens 1M							
metrics	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20
ItemPop	.181	.138	.102	.251	.163	.195	.254	.292	.157	.121	.076	.197	.154	.181	.252	.297
BPR	.348	.236	.116	.287	.370	.380	.556	.574	.341	.252	.077	.208	.349	.362	.537	.556
FISM	.426	.285	.140	.353	.462	.429	.674	.685	.420	.302	.107	.270	.443	.399	.637	.651
CDAE	.433	.287	.144	.353	.465	.425	.664	.674	.419	.307	.108	.272	.439	.401	.629	.644
GraphGAN	.212	.151	.102	.260	.183	.249	.282	.312	.178	.194	.070	.179	.205	.184	.281	.316
IRGAN	.312	.221	.107	.275	.342	.368	.536	.523	.263	.214	.072	.166	.264	.246	.301	.338
Ours	**.444**	**.294**	**.152**	**.360**	**.476**	**.433**	**.681**	**.693**	**.432**	**.309**	**.108**	**.272**	**.455**	**.406**	**.647**	**.660**

[6] Yunjey Choi et al. 2017. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. *arXiv preprint arXiv:1711.09020* (2017).
[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *ACM Recsys*. 39–46.
[8] Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Synthesizing Audio with Generative Adversarial Networks. *arXiv preprint arXiv:1802.04208* (2018).
[9] Maayan Frid-Adar et al. 2018. GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification. *arXiv preprint arXiv:1803.01229* (2018).
[10] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[11] Ian Goodfellow et al. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.
[12] Ishaan Gulrajani et al. 2017. Improved training of wasserstein gans. In *NIPS*. 5769–5779.
[13] Xiangnan He et al. 2017. Neural collaborative filtering. In *ACM WWW*. 173–182.
[14] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *IEEE ICDM*. 263–272.
[16] Cong Phuoc Huynh et al. 2018. CRAFT: Complementary Recommendations Using Adversarial Feature Transformer. *arXiv preprint arXiv:1804.10871* (2018).
[17] Won-Seok Hwang et al. 2016. "Told you i didn't like it": Exploiting uninteresting items for effective collaborative filtering. In *IEEE ICDE*. 349–360.
[18] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *ACM SIGKDD*. 659–667.
[19] Wang-Cheng Kang et al. 2017. Visually-Aware Fashion Recommendation and Design with Generative Image Models. *arXiv preprint arXiv:1711.02231* (2017).
[20] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM SIGKDD*. 426–434.
[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[23] Hung-yi Lee and Yu Tsao. 2018. Generative Adversarial Network and its Applications to Speech Signal and Natural Language Processing. (2018).
[24] Yeon-Chang Lee, Sang-Wook Kim, and Dongwon Lee. 2018. gOCCF: Graph-theoretic one-class collaborative filtering based on uninteresting items. In *AAAI*. 3448–3456.
[25] Jongwuk Lee et al. 2016. Improving the accuracy of top-N recommendation using a preference model. *Information Sciences* 348 (2016), 290–304.
[26] Youngnam Lee et al. 2018. How to impute missing ratings?: Claims, solution, and its application to collaborative filtering. In *ACM WWW*. 783–792.
[27] Xudong Mao et al. 2017. Least squares generative adversarial networks. In *IEEE ICCV*. 2813–2821.
[28] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
[29] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*. 1257–1264.
[30] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *IEEE ICDM*. 497–506.
[31] Rong Pan et al. 2008. One-class collaborative filtering. In *IEEE ICDM*. 502–511.
[32] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).
[33] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *ACM WSDM*. 273–282.
[34] Steffen Rendle et al. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[35] Suvash Sedhain et al. 2015. Autorec: Autoencoders meet collaborative filtering. In *ACM WWW*. 111–112.
[36] Yue Shi et al. 2012. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *ACM RecSys*. 139–146.
[37] Yong-Siang Shih et al. 2017. Compatibility family learning for item recommendation and generation. *arXiv preprint arXiv:1712.01262* (2017).
[38] Sumit Sidana et al. 2017. Representation learning and pairwise ranking for implicit feedback in recommendation systems. *arXiv preprint arXiv:1705.00105* (2017).
[39] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *ACM WSDM*. 93–102.
[40] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *ACM SIGKDD*. 1235–1244.
[41] Hongwei Wang et al. 2018. GraphGAN: Graph representation learning with generative adversarial nets. In *AAAI*. 2508–2515.
[42] Jun Wang et al. 2017. IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In *ACM SIGIR*. 515–524.
[43] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. Springer, 5–32.
[44] Yao Wu et al. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *ACM WSDM*. 153–162.
[45] Lantao Yu et al. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient.. In *AAAI*. 2852–2858.
[46] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering. In *ICML*. 764–773.