A collaborative filtering recommendation framework based on Wasserstein GAN

Rui Li\textsuperscript{1,a}, Fulan Qian\textsuperscript{2,b*}, Xiuquan Du\textsuperscript{3,c}, Shu Zhao\textsuperscript{4,d}, Yanping Zhang\textsuperscript{5,e}

\textsuperscript{1,2,3,4,5}School of Computer Science and Technology Anhui University, Hefei, China
\textsuperscript{a}871762980@qq.com, \textsuperscript{b*}Corresponding author: qianfulan@hotmail.com,
\textsuperscript{c}dxqlp@163.com, \textsuperscript{d}zhaoshuzs2002@163.com, \textsuperscript{e}zhangyp2@gmail.com

Abstract—Compared with the original GAN, Wasserstein GAN minimizes the Wasserstein Distance between the generative distribution and the real distribution, can well capture the potential distribution of data and has achieved excellent results in image generation. However, the exploration of Wasserstein GAN on recommendation systems has received relatively less scrutiny. In this paper, we propose a collaborative filtering recommendation framework based on Wasserstein GAN called CFWGAN to improve recommendation accuracy. By learning the real user distribution, we can mine the potential nonlinear interactions between users and items, and capture users' preferences for all items. Besides, we combine two positive and negative item sampling methods and add the reconstruction loss to the generator's loss. This can well handle the problem of discrete data in recommendation (relative to the continuity of image data). By continuously approximating the generative distribution to the real user distribution, we can finally obtain better users’ preference information and provide higher accuracy in recommendation. We evaluate the CFWGAN model on three real-world datasets, and the empirical results show that our method is competitive with or superior to state-of-the-art approaches on the benchmark top-N recommendation task.

1. INTRODUCTION

Nowadays, recommendation systems [4] have become irreplaceable tools in the network world. These systems can not only help users filter a large amount of information, but also enhance the overall experience for many consumers. Most existing recommendation systems rely on users’ feedback (purchase, rating, or review) to predict recommendations for each user. Usually, feedback information is presented in the form of a user purchase matrix. Users’ feedback can be explicit or implicit. Explicit feedback usually provides a score of 1 to 5, however, implicit feedback is generally expressed as 1 instead of distinguishing score rank. In the real world, implicit feedbacks are often easier to collect [1, 2]. In this article, we only consider implicit feedback.

Collaborative filtering [3, 5, 7] is one of the most successful techniques in the recommendation systems. This method assumes that users with similar behaviors will exhibit similar preferences for items. Matrix factorization [6, 8, 9] is a representative type of collaborative filtering algorithm which project user and item into a K-dimension latent space and directly embeds user/item ID as a vector. Then user-item interaction is modeled as the inner product of their latent vectors. The matrix factorization learns the linear interactions between latent features of users and items. With the popularity of deep learning [20, 21], many DNN-based collaborative filtering (CF) methods [10, 11, 26]
have received more and more attention and also shown their superior performance. These methods take advantage of DNN to discover nonlinear interactions between users and items.

Recently, Goodfellow et al. [15] proposed Generative Adversarial Network, one of the variants of DNN. It learns the distribution of the given data through the mini-max game between generator and discriminator. It has achieved great success in areas such as image and music generation. IRGAN [13] and GraphGAN [14] firstly try to introduce GAN into the recommendation systems and prove the feasibility and potential of GAN in the recommendation systems. Both of these two methods have the problem of “label confusion”, that is, the same item will be labeled with different positive and negative labels at the same time. This problem leads to a decline in the accuracy of the entire model. CFGAN [12] suggests a new direction, real-valued “vector-wise” adversarial training to solve this problem. That is, the model directly deals with the user’s real purchase vector instead of the user-item pair.

The original GAN uses KL divergence or JS divergence to measure the distance between distributions is not reasonable [19], while Wasserstein GAN [16] uses Wasserstein distance [27, 28] to effectively avoid such problems. Besides the Wasserstein distance used by Wasserstein GAN is also more suitable for high-dimensional and sparse recommendation data than the KL or JS divergence of the original [22, 23]. In this paper, we propose to employ Wasserstein GAN to mine potential nonlinear interactions between users and items and capture a better user’s preference information for items and ultimately improve recommendation accuracy. To solve the problem of discrete recommendation data, we combine two positive and negative item sampling methods and added the reconstruction loss to the generator's loss.

The main contributions of this paper are summarized as follows:

• We propose a collaborative filtering recommendation framework based on Wasserstein GAN, which can well capture users’ preference information and provide higher accuracy in recommendation.

• To deal with discrete recommendation data, we combine two positive and negative item sampling methods and add the reconstruction loss to the generator's loss.

• We conduct extensive experiments on three real-world datasets to demonstrate the effectiveness and rationality of the proposed CFWGAN framework.

The rest of the paper is organized as follows: In Section II, we present the related work and preliminaries. In Section III, we detailed how our framework works. And we specifically describe how our model combines two positive and negative item sampling methods to obtain user preference information. Then, in Section IV, we report the results of our extensive experiments and detailed analysis. In Section V, we finally summarize the paper.

2. RELATED WORK AND PRELIMINARIES

In this section, we first briefly introduce the research fields related to our work, including GAN-based collaborative filtering. Then we introduce the relevant reserve knowledge about Wasserstein GAN.

2.1. IRGAN and GraphGAN

IRGAN was the first paper to apply GAN to recommendation systems. For a given user, generator (G) tries to generate the items that the user may purchase, discriminator (D) tries to distinguish the user’s ground-truth items from those sampled by G. Through the mini-max game between D and G, the model can learn the embedding vectors of users and items, it can also be regarded as a special matrix factorization. Because the sampling of items is discrete each time, the policy-gradient based reinforcement learning [17] is employed to guide G by using D’s output as a reward signal to G. The ideas of GraphGAN [14] is very similar to IRGAN. GraphGAN proposes a “graph softmax” method to calculate the user’s preference for all items rather than the global calculation like IRGAN. Although IRGAN and GraphGAN were developed independently, the idea they use for CF are very similar to each other. Here, we show the objective function of IRGAN as follow:

\[
V(G, D) = \sum_u \left( E_{\times \sim \mathcal{P}(u)} [\log D(i|u)] + E_{\times \sim \mathcal{P}(i|u)} [\log (1 - D(i|u))] \right)
\]  

(1)
$P$ and $P'$ in (1) represent the real distribution and the generative distribution, respectively. $i$ and $i'$ represent the positive item and negative item that sampled from these two distributions respectively. $D(u) = \sigma(f(u, i)) = \frac{\exp(f(u, i))}{1 + \exp(f(u, i))}$, $f(u, i) = d^T_i d + b_i$ and $d_i$ are D’s latent vectors for user $u$ and item $i$, $b_i$ denotes the bias vector.

2.2. CFGAN

CFGAN points out the “label confusion” problem in GraphGAN and IRGAN. That is, the same item will be labeled with different positive and negative labels at the same time. This will reduce the efficiency of the entire model, the accuracy of recommendations will also decrease. CFGAN proposes a “vector-wise” training method to handle this. Specifically, for a given user, the generator directly generates its corresponding purchase vector. After the model is trained, the items at the position corresponding to the larger value in the fake purchase vector are recommended to the user. The objective functions of the generator and the discriminator, denoted as $J^G$ and $J^D$, are as follows:

$$J^G = \sum_u \log D(r_u | e_u) + \log(1 - D(r'_{u} \oplus e_u | e_u))$$  \hspace{1cm} (2)

$$J^D = \sum_u \log(1 - D(r'_{u} \oplus e_u | e_u))$$  \hspace{1cm} (3)

$u$ in (2) and (3) denotes user, $r_u$ and $r'_{u}$ denotes real and fake purchase vector respectively. $\oplus$ stands for element-wise multiplication. $e_u$ denotes the condition of each user, $e_u$ is an indicator vector.

2.3. WassersteinGAN

For the original GAN, when the discriminator is optimal, the generator’s loss is equivalent to minimizing the JS-divergence between the real distribution and the generative distribution. However, since these two distributions have almost no overlap or negligible overlap, the JS-divergence between them is a constant log2 regardless of how far apart they are, and eventually, the G’s gradient will be approximately 0. Wasserstein GAN proposes to use Wasserstein distance instead of JS-divergence to measure the distance between two distributions, solves the problem of gradient disappearance, and makes GAN’s training more stable. The specific approach can be summarized as: The last layer of the D removes the sigmoid activation function. The loss functions of the G and D do not take $\log$. Each time the parameters of D are updated, their absolute values are truncated to no more than a fixed constant $c$. Formally, the objective function of Wasserstein GAN can be express as $V$:

$$\max_{\theta} \min_{\phi} V(G,D) = E_{x \sim P_{data}} D(x) - E_{z \sim P_{z}} D(G(z))$$  \hspace{1cm} (4)

In (4), $x$ is a ground-truth data sampled from the real distribution $P_{data}$, $x'$ is a fake data sampled from the generative distribution $P_{z}$ which is generated by G. $D(\cdot)$ indicates the estimated probability of its input being a ground-truth data.

3. CFWGAN

In this section, we first define the notations to be used. Next, we present an overview of the proposed CFWGAN framework. Finally, we introduce two positive and negative item sampling methods and specifically explain how to reconstruct the generator loss based on the items we sampled.

3.1. Notations

$U = \{u_1, u_2, ..., u_m\}$ and $I = \{i_1, i_2, ..., i_n\}$ denote a set of m users and a set of n items. We define $I_u$ as a set of items receiving implicit feedback from $u$. User’s implicit feedback on items is represented by a sparse matrix $R = (r_u)_{m \times n}$. If $i$ in $I_u$, then $r_{ui}$, otherwise it is 0. For each user $u$, we denote the real purchase vector as $r_u$, which is an n-dimensional vector, corresponding to the vector in the row $u$ of the matrix $R$. 
3.2. The framework of CFWGAN
The element 1 in the \( r_u \) represents the user’s interaction with the items and reflects the user’s preference information for the items. But this kind of preference information is only part of what we can obtain. The element 0 in the \( r_u \) does not necessarily mean that the user does not like the item, it may be because the user is not aware of the item. Our goal is to reconstruct the complete user’s preference information based on the known partial user preference information. Then recommend items to users based on the complete user preferences information.

We employ Wasserstein GAN as our basic framework. Fig. 1 shows an overview of the model. For a given \( r_u \), our goal is to employ Wasserstein GAN to learn the potential nonlinear interaction information between the user and the item. The generator(G) reconstructs the complete user’s preference information according to the learned interaction information and expresses it in the form of a vector, which we record as the user’s fake purchase vector \( r_u' \). \( r_u' \) is also an n-dimension vector which is expected to be a sparse vector where all elements corresponding to the purchase items of \( u \), \( r_u' \) where \( i \in I_u \) are hopefully 1. The value of the element in the \( r_u' \) represents the user's preference for the corresponding item. The discriminator (D) is responsible for identifying the real purchase vector and the fake purchase vector. A higher score is given to the real purchase vector, and a lower score is given to the fake purchase vector. Through the min-max game between D and G, the generative distribution is gradually pushed to the real user distribution. G can mine the potential nonlinear interactions between users and items, and capture users’ preferences for all items. D can also discriminate between these two kinds of vectors well. Eventually, D and G can reach a Nash equilibrium.

We follow Wasserstein GAN, the objective function of D denoted as \( J^D \) is defined as follows:

\[
J^D = E_{r_u \sim \mu_d} D(r_u) - E_{r_u' \sim \mu_g} D(r_u') = \sum_i D(\rho_u' | r_u) - \sum_i D(\rho_u | r_u) \tag{5}
\]

Similarly, the objective function of G denoted as \( J^G \) is defined as follow:

\[
J^G = -\sum_i D(\rho_u' | r_u) \tag{6}
\]

Equations (5) and (6) represent the objective functions of D and G, respectively. \( \rho_u \) and \( \rho_u' \) represent generative distribution and real user distribution, respectively. \( r_u' \) and \( r_u \) represent fake purchase vector and real purchase vector of the user \( u \). In the actual process, we concatenate \( r_u' \) and \( r_u \) with their real purchase vectors \( r_r \) before sending them to D. The concatenate vector can be regarded as the condition of each user, such processing can ensure that each user’s purchase vector can be distinguished from
others. We implement both G and D as n-layer neural networks. The formula for forward propagation of G and D can be expressed as follow:

\[ y = \sigma(w \cdot x + b) \]  

(7)

\( w \) and \( b \) in (7) are the weight matrix and bias vector, respectively. \( x \) denotes the input of the current layer. \( \sigma \) denotes activation function, here we use sigmoid. We follow Wasserstein GAN to remove the activation function of the last layer of D.

For G, its input is a batch of \( r^\prime \), and its output is a batch of \( r^\prime_i \). The goal of G is to make \( r^\prime \) and \( r^\prime_i \) as similar as possible. For D, its input is a batch of \( r_i \) or \( r^\prime_i \) concatenate with their real purchase vector \( r_i \). Its output a single scalar value which denotes the probability that its input came from the ground-truth data, rather than G. The goal of D is to distinguish these two vectors as much as possible.

3.3. Two sampling methods

The image is often represented as a dense vector composed of pixels with multiple bits depth while the purchase vector is always some high dimensional and sparse. Without any processing, it is unreasonable to directly apply Wasserstein GAN to the recommendation data. The red curve in Fig. 3 shows the iteration result without adding any constraints to the model. It can be seen that the recommendation effect is very poor. To well handle the discrete recommendation data, we consider adding constraints to the generator of the original CFWGAN model. The specific approach is: we sample a part of items and calculate the reconstruction loss of the corresponding position in the fake purchase vector, and then add it to the G’s loss. The sampling methods can be divided into the following two categories:

3.3.1. negative item sampling and ZR loss

In every training iteration, we randomly sample a portion of the user’s non-purchased items (\( I^\prime \)) and presume them as negative items (The red part in Fig. 1), indicating their corresponding feedback in matrix R is not missing but zero. Then, we train G to generate a fake purchase vector in such a way that the values on the negative items are close to 0. It can be understood that if the elements in the fake purchase vector corresponding to the position of negative items are not 0, the loss of G will increase. We use N-ZR to denotes a set of these sampled negative items and S-ZR denotes the percentage of negative items (For example, if S-ZR=50, then we sample 50 percent items in \( I^\prime \)). We expand the objective function in (6). The objective function of G can be expressed as follow:

\[ J^G = \sum \left(-D(r^\prime_i) \right) + \alpha \sum (x^r_i - x^\prime_i \gamma) \]  

(8)

\( \sum \) denotes the accumulation of loss for all users. The first term in \( \sum \) is the score given by D. The second term denotes zero reconstruction (ZR) loss where \( i \in N-ZR \), \( \alpha \) is a tunable parameter for controlling the importance of the ZR loss in the entire equation.

3.3.2. positive item sampling and OR loss

Similarly, we randomly sample a portion of each user’s purchased items (\( I \)) and presume them as positive items (The green part in Fig. 1). We train G to generate a fake purchase vector in such a way that the values on the positive items are close to 1. By doing this, we hope that the elements in the fake purchase vector corresponding to the position of positive items to be 1 as much as possible. Hence, G not only concentrate on generating the fake purchase vector to deceive D, but also consider generating values close to 1 on the positive items. N-OR denotes a set of these selected negative items and S-OR denotes the percentage of positive items. The objective function of G can be expressed as follow:

\[ J^G = \sum \left(-D(r^\prime_i) \right) + \beta \sum (x^r_i - x^\prime_i \gamma) \]  

(9)

In (9), the second term in \( \sum \) denotes one reconstruction (OR) loss where \( i \in N-OR \), \( \beta \) is also a tunable parameter for controlling the importance of the OR loss in the entire equation.
In the implementation process, we combine these two sampling methods. The hybrid method uses both OR loss and ZR loss (OZ). Both positive and negative items contribute to the learning process of G. The objective function of G can be expressed as follow:

$$J^G = \sum_i (-D(r^*_i, r_i)) + \alpha \sum_i (x^*_i - x_i)^2 + \beta \sum_j (x^*_j - x^*_j)^2$$

In (10), the second and third terms in $\sum(\cdot)$ denote ZR loss and OR loss, respectively.

4. EXPERIMENTS

4.1. Experimental settings

4.1.1. Datasets

We conduct our experiments on three representative real-world datasets. Table I summarizes the statistics of datasets Ciao, Movielens 100K, and Movielens 1M. For each dataset, we randomly split its user-item interactions into two subsets: 80% for training and the rest 20% for testing.

| Datasets    | #user | #item | #interaction | Sparsity   |
|-------------|-------|-------|--------------|------------|
| Ciao        | 996   | 1,927 | 18,648       | 98.72%     |
| Movielens 100K | 943   | 1,682 | 10,000       | 93.69%     |
| Movielens 1M | 6,039 | 3,883 | 1,000,209    | 95.72%     |

4.1.2. Baselines

- **ItemPop** It is a non-personalized algorithm that items are recommended based on how many users have rated them.
- **BPR**[24] This is an MF-based model recommendation for implicit feedback problems. It optimizes MF with the BPR objective.
- **FISM**[25] It is a collaborative filtering algorithm based on items. It’s also an algorithm based on implicit feedback.
- **CDAE**[26] CDAE is based on denoising autoencoder. The input data is corrupted by noise before fed into the neural network.
- **IRGAN**[13] It is the first GAN-based CF method whose details are already described in Section II-A.
- **GraphGAN**[14] This method combines generative and discriminative information retrieval via adversarial training. The author proposes a “graph softmax” for the generator to solve the problem of much calculation.
- **CFGAN**[12] This is a collaborative filtering algorithm based on conditional GAN. It suggests a new direction, real-valued “vector-wise” adversarial training.

![Graph showing precision@5 for Ciao and Movielens 100K](image-url)
4.1.3. Evaluation metrics
We use four popular accuracy metrics for top-N recommendations: recall (R@N), nDCG (G@N), MRR (M@N), and precision (P@N). We set N as 5 and 20.

4.1.4. Implementation details
We use sigmoid function as an activation function for neural networks and the Xavier’s approach [18] for initializing the hidden layer weight matrix. We search the number of hidden layers of D and G in {1,2,3,4}, the number of hidden nodes per hidden layer in {50,100,150,200,300}. The learning rate is tuned amongst {0.005,0.001,0.0005}. The minibatch is tuned amongst {32,64,128,256}. S-ZR and S-OR are tuned in {30,40,50,60,70,80,90}, \( \alpha \) and \( \beta \) are tuned in {0.5,0.2,0.1,0.05,0.03,0.01}.

4.2. Experimental analysis
Fig. 3 shows the training curves of different sampling methods. Through the curves, we can find that when we only use Wasserstein GAN without adding additional constraints to recommend, the effect is not very good. When we add ZR and OR constraints separately, the accuracy of the model is significantly improved, which proves that the constraints we added are useful. When we add both of ZR loss and OR loss, we combine ZR loss and OR loss, we find that the accuracy of the model is higher, and the curve tends to be smooth. We believe that both positive and negative items contribute to the final recommendation result.

We draw the comparison curve between CFWGAN and our competitive baseline CFGAN. We set the parameters of the CFGAN algorithm to the optimal parameters mentioned in the original paper. As
shown in Fig. 2 that our iterative curve converges faster and the curve is smoother. Table II, Table III, and Table IV report the comparison results for all the baselines on the three datasets. For the Ciao dataset, when we set N to 20, CFWGAN outperforms all the baselines on the four chosen metrics. When we set N as 5, the competitive baseline

![Fig. 4 Accuracy depending on S-ZR and S-OR](image1)

![Fig. 5 Accuracy depending on \( \alpha \) and \( \beta \)](image2)

CFGAN and CFWGAN can achieve similar results. It may be that the data set is too sparse, causing the performance of GAN to not fully play out. For the Movielens 100K dataset, compared with GAN-based methods: GraphGAN, IRGAN, CFGAN, the accuracy was improved up to 113.7%, 45.2%, 2.0% in terms of P@5, respectively. The accuracy of nDCG and MRR and recall has also improved significantly. Movielens 1M is a relatively large dataset, and our method can also achieve better results. Compared with competitive baseline CFGAN, the accuracy of CFWGAN is improved by 2.1%, 3.7%, 1.8%, and 1.2% in terms of P@5, R@5, G@5, and M@5, respectively.

4.3. Parameter analysis
All the hyper-parameters used in our method are shown in Section IV-A-4). Here, we investigate the influence of the following unique parameters of our method on the experimental results: S-ZR, S-OR, \( \alpha \), and \( \beta \). Fig. 4 shows the accuracy depending on S-ZR and S-OR. When we set S-OR above 80, the effect of the model changes little. And when we set S-ZR to 40∼50 the model tends to perform well. Too much or too little sampling of negative items will constrain the performance of the model. In the actual process, we set S-ZR=40, S-OR=90, the model can often get satisfactory results. This may be a good balance between positive and negative items. Fig. 5 shows the accuracy depending on \( \alpha \) and \( \beta \). We observe that the moderate range of and values would be in [0.03,0.05] can often achieve better results. We believe that it is a good balance, which can make ZR and OR terms fully function.
5. CONCLUSIONS
In this paper, we employ Wasserstein GAN to mine potential nonlinear interactions between users and items to capture a better user’s preference information. Besides, for discrete recommendation data, we propose two sampling methods to make the obtained user’s preference more realistic. Therefore, the accuracy of the recommendation is improved.

| TABLE II. COMPARISON RESULTS IN CIAO |
|-------------------------------------|
| metrics | P@5 | P@20 | R@5 | R@20 | G@5 | G@20 | M@5 | M@20 |
|---------|------|------|-----|------|-----|------|-----|------|
| ItemPop | 0.031 | 0.024 | 0.040 | 0.127 | 0.047 | 0.065 | 0.056 | 0.067 |
| BPR     | 0.036 | 0.025 | 0.040 | 0.141 | 0.052 | 0.066 | 0.066 | 0.078 |
| FISM    | 0.062 | 0.040 | 0.072 | 0.178 | 0.079 | 0.109 | 0.127 | 0.147 |
| CDAE    | 0.061 | 0.042 | 0.075 | 0.185 | 0.081 | 0.108 | 0.127 | 0.151 |
| GrapGAN | 0.026 | 0.017 | 0.041 | 0.100 | 0.041 | 0.058 | 0.057 | 0.068 |
| IRGAN   | 0.035 | 0.023 | 0.042 | 0.111 | 0.046 | 0.066 | 0.082 | 0.088 |
| Ours    | 0.071 | 0.046 | 0.082 | 0.199 | 0.092 | 0.126 | 0.152 | 0.177 |

| TABLE III. COMPARISON RESULTS IN MOVIELENS 100K |
|-----------------------------------------------|
| metrics | P@5 | P@20 | R@5 | R@20 | G@5 | G@20 | M@5 | M@20 |
|---------|------|------|-----|------|-----|------|-----|------|
| ItemPop | 0.181 | 0.138 | 0.102 | 0.251 | 0.163 | 0.195 | 0.254 | 0.292 |
| BPR     | 0.348 | 0.236 | 0.116 | 0.287 | 0.370 | 0.380 | 0.556 | 0.574 |
| FISM    | 0.426 | 0.285 | 0.140 | 0.353 | 0.462 | 0.429 | 0.674 | 0.685 |
| CDAE    | 0.433 | 0.287 | 0.144 | 0.353 | 0.465 | 0.425 | 0.664 | 0.674 |
| GrapGAN | 0.212 | 0.151 | 0.102 | 0.260 | 0.183 | 0.249 | 0.282 | 0.312 |
| IRGAN   | 0.312 | 0.221 | 0.107 | 0.275 | 0.342 | 0.368 | 0.536 | 0.523 |
| CFGAN   | 0.444 | 0.294 | 0.152 | 0.360 | 0.476 | 0.433 | 0.681 | 0.693 |
| Ours    | 0.453 | 0.300 | 0.158 | 0.362 | 0.486 | 0.442 | 0.694 | 0.702 |

| TABLE IV. COMPARISON RESULTS IN MOVIELENS 1M |
|---------------------------------------------|
| metrics | P@5 | P@20 | R@5 | R@20 | G@5 | G@20 | M@5 | M@20 |
|---------|------|------|-----|------|-----|------|-----|------|
| ItemPop | 0.157 | 0.121 | 0.076 | 0.197 | 0.154 | 0.181 | 0.252 | 0.297 |
| BPR     | 0.341 | 0.252 | 0.077 | 0.208 | 0.349 | 0.362 | 0.537 | 0.556 |
| FISM    | 0.420 | 0.302 | 0.107 | 0.270 | 0.443 | 0.399 | 0.637 | 0.651 |
| CDAE    | 0.419 | 0.307 | 0.108 | 0.272 | 0.439 | 0.401 | 0.629 | 0.644 |
| GrapGAN | 0.178 | 0.194 | 0.070 | 0.179 | 0.205 | 0.184 | 0.281 | 0.316 |
| IRGAN   | 0.263 | 0.214 | 0.072 | 0.166 | 0.246 | 0.246 | 0.301 | 0.338 |
| CFGAN   | 0.432 | 0.309 | 0.108 | 0.272 | 0.455 | 0.406 | 0.647 | 0.660 |
| Ours    | 0.441 | 0.316 | 0.112 | 0.278 | 0.463 | 0.415 | 0.655 | 0.669 |

ACKNOWLEDGMENT
This work was supported by the Youth Program of National Natural Science Foundation of China (No.61702003), the Natural Science Foundation of Anhui Province of China (No.61702003), and the National Natural Science Foundation of China (No.61673020, No.61876001).

REFERENCES
[1] Hu Y, Koren Y, Volinsky C. Collaborative filtering for implicit feedback datasets[C]//2008 Eighth IEEE International Conference on Data Mining. Ieee, 2008: 263-272.
[2] Pan R, Zhou Y, Cao B, et al. One-class collaborative filtering[C]//2008 Eighth IEEE International Conference on Data Mining. IEEE, 2008: 502-511.
[3] Schafer J B, Frankowski D, Herlocker J, et al. Collaborative filtering recommender systems[M]//The adaptive web.Springer, Berlin, Heidelberg, 2007: 291-324.
[4] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. IEEE transactions on knowledge and data engineering, 2005, 17(6): 734-749.

[5] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C]//Proceedings of the 10th international conference on World Wide Web. 2001: 285-295.

[6] He X, Zhang H, Kan M Y, et al. Fast matrix factorization for online recommendation with implicit feedback[C]//Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. 2016: 549-558.

[7] He X, Liao L, Zhang H, et al. Neural collaborative filtering[C]//Proceedings of the 26th international conference on world wide web. 2017: 173-182.

[8] Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model[C]//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008: 426-434.

[9] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems[J]. Computer, 2009, 42(8): 30-37.

[10] Wang H, Wang N, Yeung D Y. Collaborative deep learning for recommender systems[C]//Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 2015: 1235-1244.

[11] Sedhain S, Menon A K, Sanner S, et al. Autorec: Autoencoders meet collaborative filtering[C]//Proceedings of the 24th international conference on World Wide Web. 2015: 111-112.

[12] Chae D K, Kang J S, Kim S W, et al. Cfgan: A generic collaborative filtering framework based on generative adversarial networks[C]//Proceedings of the 27th ACM international conference on information and knowledge management.2018:137-146.

[13] He Wang J, Yu L, Zhang W, et al. Irgan: A minimax game for unifying generative and discriminative information retrieval models[C]//Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 2017: 515-524.2.

[14] Wang H, Wang J, Wang J, et al. Graphgan: Graph representation learning with generative adversarial nets[C]//Thirty-second AAAI conference on artificial intelligence. 2018.

[15] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.

[16] Arjovsky M, Chintala S, Bottou L. Wasserstein gan[J]. arXiv preprint arXiv:1701.07875, 2017.

[17] Yu L, Zhang W, Wang J, et al. Seqgan: Sequence generative adversarial nets with policy gradient[C]//Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[18] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In AISTATS. 249–256

[19] Arjovsky M, Bottou L. Towards principled methods for training generative adversarial networks[J]. arXiv preprint arXiv:1701.04862, 2017.

[20] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436-444.

[21] Zhang S, Yao L, Sun A, et al. Deep learning based recommender system: A survey and new perspectives[J]. ACM Computing Surveys (CSUR), 2019, 52(1): 1-38.

[22] Liu H, Wen J, Jing L, et al. Deep generative ranking for personalized recommendation[C]//Proceedings of the 13th ACM Conference on Recommender Systems. 2019: 34-42.

[23] Krishnan R, Liang D, Hoffman M. On the challenges of learning with inference networks on sparse, high-dimensional data[C]//International Conference on Artificial Intelligence and Statistics. 2018: 143-151.

[24] Rendle S, Freudenthaler C, Gantner Z, et al. BPR: Bayesian personalized ranking from implicit feedback[J]. arXiv preprint arXiv:1205.2618, 2012.
[25] Kabbur S, Ning X, Karypis G. Fism: factored item similarity models for top-n recommender systems[C]//Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013: 659-667.

[26] Wu Y, DuBois C, Zheng A X, et al. Collaborative denoising auto-encoders for top-n recommender systems[C]//Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. 2016: 153-162.

[27] Y. Rubner. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval[C]// Proceedings of the Arpa Image Understanding Workshop. Piscataway: IEEE, 1997: 661-668.

[28] Y. Rubner, C. Tomasi, L.J. Guibas. The earth mover's distance as a metric for image retrieval[J]. International Journal of Computer Vision. 2000, 40(2): 99-121.