LambdaGAN: Generative Adversarial Nets for Recommendation Task with Lambda Strategy

Yang Wang
Tsinghua-Southampton Web Science Laboratory
Graduate School at Shenzhen, Tsinghua University
Shenzhen, China
yang-wan17@mails.tsinghua.edu.cn

Hai-Tao Zheng*
Tsinghua-Southampton Web Science Laboratory
Graduate School at Shenzhen, Tsinghua University
Shenzhen, China
zheng.haitao@sz.tsinghua.edu.cn

Wang Chen
Tsinghua-Southampton Web Science Laboratory
Graduate School at Shenzhen, Tsinghua University
Shenzhen, China
chen-w16@mails.tsinghua.edu.cn

Rui Zhang
School of Computing and Information Systems
University of Melbourne
Melbourne, Australia
rui.zhang@unimelb.edu.au

***Abstract*—The Top-N recommendation task aims to recommend users the items they like most. Generative Adversarial Net (GAN) has achieved good results on recommendation, which learns user-item matrix by a generative adversarial training process. There are two main scenarios, pointwise scenarios and pairwise scenarios. However, for recommendation, GANs in pairwise scenarios perform not as well as these in pointwise scenarios. As pairwise rank is a position-independent algorithm, it does not consider Top-N ranking sufficiently. Recommendation task is position-dependent. Especially the Top-N item ranking accuracy is much more important than the ranking accuracy of the tail item. In this paper, we propose LambdaGAN for Top-N recommendation. LambdaGAN introduces lambda rank into generative adversarial training process in order to consider the ranking information of the item. The proposed model enables generative adversarial training in pairwise scenarios available for recommendation by optimizing the rank based metrics directly. Moreover, we adjust lambda function according to the characteristics of recommendation. Two new designed lambda functions are proposed. Experimental results show that LambdaGAN outperforms state-of-the-art algorithms including BPR, PRFM, LambdaFM and IRGAN in terms of four standard evaluation metrics on two widely used datasets, Movielens-100K and Netflix.**

***Index Terms*—Top-N Recommendation, Generative Adversarial Training, Pairwise Rank**

I. Introduction

Recommendation System (RS) is designed to recommend users the items they may like. Because of the huge amount of items in the recommendation task, the Top-N tasks have more practical significance, which can provide users with the items they like most [1].

In the traditional recommendation model, LambdaFM is one of the state-of-the-art models [2]. Based on [3], [4], LambdaFM combines lambda rank [5] with Factorization Machine (FM) [4] and optimizes the rank based metrics directly. That makes it very effective in Top-N recommendation. Although a series of recommendation models represented by LambdaFM have achieved impressive results, the above methods are all trained by a single traditional FM model which can not achieve optimal performance sometimes.

At the same time, the excellent performance of deep learning also provides a better way to further improve the performance of recommendation models. Information Retrieval Generative and Discriminative model (IRGAN) [6] unifies traditional generative model and discriminative model into a minimax game in the information retrieval task. These two models' performance gets improved by the process of adversarial learning. IRGAN has achieved very excellent experimental results in recommendation. However, it is worth noticing that, GANs [8] for recommendation in pairwise scenarios perform not as good as these in pointwise scenarios. The reason is that pairwise rank does not perform well in the recommendation task. Because pairwise rank is position-independent, an incorrect pairwise item pair at the bottom of the list and the one at the top of the list are equally important for pairwise scores [2]. In other words, pairwise loss is not the optimal for the Top-N recommendation task.

In this paper, we propose LambdaGAN for Top-N recommendation. The proposed model applies lambda strategy into generative adversarial training. And our model is optimized by the rank based metrics directly. So we can make generative adversarial training in pairwise scenarios available for recommendation. In addition, we rewrite the original lambda functions to suit the characteristics of recommendation. Two new designed lambda function are proposed. Our work and contributions are mainly listed as follows:

- We are the first one to combine lambda strategy with generative adversarial learning in pairwise scenarios, and apply our model (LambdaGAN) to Top-N recommendation. This combination can improve the performance of Top-N Recommendation.

*Hai-Tao Zheng is the corresponding author.

- we explore the limitations of original lambda rank in the recommendation task and propose two new lambda functions. These two lambda functions serve as penalty and reward parameters for pairwise learning respectively.
- Experimental results show LambdaGAN significantly outperforms the other strong baseline methods including BPR [7], PRFM [3], LambdaFM and IRGAN on Moivelens-100K and Netflix in terms of four rank based evaluation metrics.

The rest of this paper is origanized as follows. In Section II, we review the related work on recommendation. In Section III, we present the adversarial learning for item recommendation and describe the lambda strategy. Two versions of LambdaGAN will be proposed. In Section IV, we present the experimental results of LambdaGAN and the other baseline methods. In Section V, we conclude our work.

II. Related Work

The method presented in this paper is mainly based on Generative Adversarial Nets(GAN) [8] and Learning to Rank(LtR) techniques. In this part we briefly review the application of these two parts in recommendation system and show the differences from our methods.

Generative Adversarial Nets for Recommendation. GAN has achieved impressive results in the field of computer vision [9]–[11], sequence generation [12] and data mining [13], [14]. The work related to us is [6]. It proposes the information retrieval model based on the generative model and discriminative model together with playing a minimax game. At the same time, impressive experimental results have been achieved on these three tasks (Web Search, Item Recommendation and Question Answering). Our work is different from IRGAN mainly in several aspects. There are two versions of IRGAN: IRGAN-pointwise and IRGAN-pairwise. However, IRGAN only performs the recommendation tasks in pointwise scenarios. One possible reason is that pairwise rank is insensitive to item positions of the ranking list, but the recommendation tasks are more concerned with the Top-N item list than the whole item ranking list. Based on the above problem, we introduce the lambdaRank into generative adversarial training process for recommendation.

Learing to Rank. There are two main methods in the field of LtR, pairwise [7], [15]–[17] and listwise methods [18]–[20] respectively. The goal of pairwise optimization is to reduce the number of wrong item pairs. It does not directly optimize the Top-N items in recommendation. Pairwise learning is a suboptimal solution for recommendation. On the other hand, evaluation metrics such as NDCG [17] and MAP [20] are discontinuous, so we cannot directly use the traditional stochastic gradient descent method for optimization. Inspired by lambda rank, we propose lambda strategy to optimize pairwise rank so that the algorithm focuses on the Top-N item list. LambdaFM is a work which uses lambda strategies to optimize PRFM in recommendation [2]. The difference between him and our work is that: first, we introduce lambda strategy into generative adversarial nets for recommendation. Second, because of the different usage scenarios, we have applied a new lambda function based on the adversarial learning.

III. Methodology

In this section, we encapsulate adversarial learning in pairwise scenarios for recommendation. Then we describe the lambda strategy and propose two versions of LambdaGAN.

A. Adversarial Learning in Pairwise Scenarios

GAN consist of a generator G and a discriminator D that compete in a minimax game with two players. The performance of G and D are improved by the feedback from each other during the training process.

Generative retrieval model $P_\theta(i|u,r)$, which tries to generate (or predict) relevant items, from the candidate pool for the given user u; in other words, its goal is to approximate the true relevance distribution over items $P_{true}(i|u,r)$ as much as possible.

Discriminative retrieval model $f_\phi(u,i)$, which, by contrast, tries to discriminate ground truth user-item tuples (u,i) from faked ones, where the goodness of matching given by $f_\phi(u,i)$ depends on the relevance of item to user; in other words, its goal is to distinguish between favourite items and non-favourite items for the user u as accurately as possible. In fact, it is a binary classifier, and we could use 1 as the class label for the user-item tuples that truly match (positive examples) while 0 as the class label for those that do not really match (negative examples).

1) Overall Objective: The discriminator tries to distinguish real high-related items on training data from recommendation list predicted by G, and the generator tries to fool the discriminator to generate (or predict) well-ranked recommendation list. Formally, we have minimax game in pointwise scenarios:

$$J^{G^*,D^*} = \min_\theta \max_\phi \sum_{n=1}^{N} (\mathbb{E}_{d\sim p_{true}(i|u_n,r)}[logD(i|u_n)+ \mathbb{E}_{d\sim p_\theta(i|u_n,r)}[log(1-D(i|u_n))]) \tag{1}$$

Where $p_\theta(i|u_n,r)$ represents the generative retrieval model and the discriminative retrieval model $D(i|u_n)$ shows the probility of the item i being selected by user u, which is given by the sigmod function of the relevance score:

$$D(i|u) = \sigma(f_\phi(u,i)) = \frac{\exp(f_\phi(u,i))}{1+\exp(f_\phi(u,i))} \tag{2}$$

In recommendation task, $f_\phi(u,i)$ is replaced by a scoring function. Collaborative filtering method is the one of the most widely used methodologies which aims to find the similarity of user to user or item to item. Therefore, our scoring function of the preference of user u to item i can be given by :

$$f_\phi(u,i) = s(u,i) = b_i + \nu_u^\top \nu_i \tag{3}$$

Where b_i is the bias term of item i, $\nu_u, \nu_i \in \mathbb{R}^k$ are the latent vectors of user u and item i defined in a k-dimensional continuous space respectively.

2) optimizing discriminative model: The objective of discriminative model D is to maximize the probability of correctly distinguishing the ground truth items from the generated recommendation items. With the ground truth items, and the items given by the current optimal generative model G, we can obtain the optimal parameters for the discriminative model :

$$\phi^* = arg\ \max_{\phi} \sum_{n=1}^{N} (\mathbb{E}_{d\sim p_{true}(i|u_n,r)})[log(\sigma(b_i + \nu_{u_n}^\top \nu_i))] + \mathbb{E}_{d\sim p_{\theta^*}(i|u_n,r)}[log(1-\sigma(b_i + \nu_{u_n}^\top \nu_i))] \tag{4}$$

The above can be solved by stochastic gradient descent.

3) optimizing Generative model: Different to discriminative model, the generative model aims to minimize the objective. It tries to fit the true distribution of items i for user u, and given the generated items from the whole item list to cheat the discriminative model. The generative model G can be optimized by minimizing the following formulation:

$$\begin{aligned}\theta^* &= arg\ \min_{\theta} \sum_{n=1}^{N} (\mathbb{E}_{d\sim p_{true}(i|u_n,r)}[log\sigma(b_i + \nu_{u_n}^\top \nu_i)] + \mathbb{E}_{d\sim p_\theta(i|u_n,r)}[log(1-\sigma(b_i + \nu_{u_n}^\top \nu_i))]) \\ &= arg\ \max_{\theta} \sum_{n=1}^{N} \underbrace{\mathbb{E}_{d\sim p_\theta(i|u_n,r)}[log(1+\exp(b_i + \nu_{u_n}^\top \nu_i)))]}_{denote\ as\ J^G(u_n)}\end{aligned} \tag{5}$$

Since the sampling of recommendation list by G is discrete, which cannot be optimized by gradient descent as in the original GAN formulation. So we use policy gradient based reinforcement learning [12], [21] to optimize the Generative model. The results are as follows:

$$\begin{aligned}&\nabla_\theta J^G(u_n) \\ &= \nabla_\theta \mathbb{E}_{d\sim p_\theta(i|u_n,r)}[log(1+\exp(b_i + \nu_{u_n}^\top \nu_i))] \\ &\simeq \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta log p_\theta(i_k|u_n,r) log(1+\exp(b_i + \nu_{u_n}^\top \nu_{i_k}))\end{aligned} \tag{6}$$

Where i_k denotes the k-th items sampled by the current version of generative model G. With reinforcement learning technology, $\log(1+\exp(b_i + \nu_{u_n}^\top \nu_{i_k}))$ acts as the reward for the policy $p_\theta(i|u_n, r)$ taking an action i in the environment u_n [22].

In pairwise scenarios, we take the positive feedback item given by users as positive sample i_p, the others in the list as negative sample i_q. Thus we have the labelled item pairs $R_n = \{\langle i_p, i_q\rangle | i_p \succ i_q\}$ where $i_p \succ i_q$ means user prefers the item i_p to the item i_q.

The generator model G aims to produce item pairs with the correct ranking. In contrast, the discriminator model D aims to distinguish real item pairs from the faked ones created by generator. The discriminator model in pairwise scenarios is rewritten as :

$$\begin{aligned}D(\langle i_p, i_q\rangle | u) &= \sigma(f_\phi(i_p, u) - f_\phi(i_q, u)) \\ &= \frac{\exp(f_\phi(i_p, u) - f_\phi(i_q, u))}{1+\exp(f_\phi(i_p, u) - f_\phi(i_q, u))}\end{aligned} \tag{7}$$

Algorithm 1 Adversarial Learning for Recommendation in Pairwise Scenarios

Input: generator $p_\theta(i|u,r)$; discriminator $f_\phi(u,i)$; training datasets S

1: Initialize $p_\theta(i|u,r)$, $f_\phi(u,i)$ with random weights θ, ϕ.
2: pre-train generator: $P_\theta(i|u,r)$ and discriminator: $f_\phi(u,i)$ using S
3: **repeat**
4: **for** g-steps **do**
5: $p_\theta(i|u,r)$ generates K possible liked items for each user u
6: Update generator parameters via policy gradient
7: **end for**
8: **for** d-steps **do**
9: Use current generator $p_\theta(i|u,r)$ to generate negative examples and combine with given positive example S to form a new training Sets $S^{'}$
10: Train discriminator $f_\phi(u,i)$ by Eq.(9)
11: **end for**
12: **until** algorithm converges

Where $f_\phi(u,i)$ is a scoring function for recommendation.

$$f_\phi(u,i) = s(u,i) = b_i + \nu_u^\top \nu_i \tag{8}$$

Then we have the following minimax game in pairwise scenarios:

$$J^{G^*,D^*} = \min_{\theta} \max_{\phi} \sum_{n=1}^{N} (\mathbb{E}_{o\sim p_{true}(\mathbf{o}|u_n)}[logD(\mathbf{o}|u_n)] + \mathbb{E}_{o'\sim p_\theta(\mathbf{o}'|u_n)}[1-logD(\mathbf{o}'|u_n)]) \tag{9}$$

where $\mathbf{o} = \langle i_p, i_q\rangle$ and $\mathbf{o}' = \langle i_p^{'} i_q^{'}\rangle$ are real and generated item pairs for user u_n respectively. $P_\theta(i|u,r)$ is the generative model and $P_{true}(i|u,r)$ is the true relevance distribution over items.

Algorithm 1 summarizes the overall process of adversarial learning in pairwise scenarios for item recommendation. We first pre-train the parameters of generator and discriminator. Then, in the processing of adversarial learning, generator and discriminator have a better performance with the implict feedback given by each other.

B. Lambda Strategy

As shown in Fig. 1, Fig. 1(b) has one more pairwise error than Fig. 1(c). However, the value of NDCG for Fig. 1(b) is 0.798 higher than 0.525 in Fig. 1(c), which means the rank list of Fig. 1(b) has a better performance than Fig. 1(c). Thus, this shows that pairwise rank is not the optimal method for the Top-N recommendation task. Based on Fig. 1(c), the black solid arrow and the red dotted one represent two ways to minimize the pairwise error in next iteration. The black arrow moves two positive items from 3rd and 6th to 2nd and 4th respectively. The red arrow moves two positive items from 3rd and 6th to 1st and 5th respectively. After these two optimizations, the red arrow increases NDCG value by 0.325, and the black arrow

Algorithm 2 Adversarial Learning for Recommendation with Lambda Strategy (LambdaGAN)

Input: generator $p_\theta(i|u,r)$; discriminator $f_\phi(u,i)$; training datasets S

Initialize $p_\theta(i|u,r), f_\phi(u,i)$ with random weights θ, ϕ.
pre-train generator: $P_\theta(i|u,r)$ and discriminator: $f_\phi(u,i)$ using S
repeat
  for g-steps **do**
    $p_\theta(i|u,r)$ generates K possible liked items for each user u
    Update generator parameters via policy gradient
  end for
  for d-steps **do**
    Use current generator $p_\theta(i|u,r)$ to generate negative examples and combine with given positive example S to form a new training Sets $S^{'}$
    Train discriminator $f_\phi(u,i)$ with lambda strategy by Eq. (11)
  end for
until LambdaGAN converges

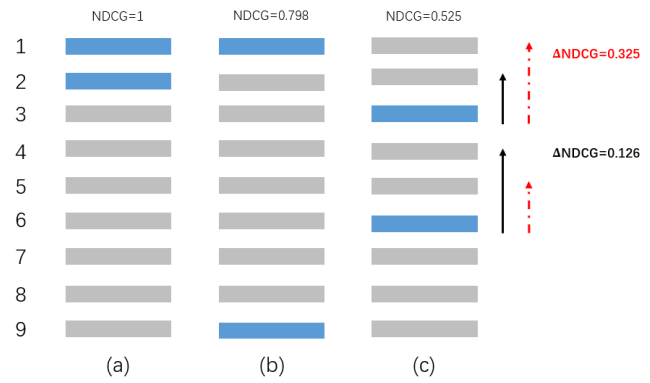


Fig. 1. Three different item list for a given user. Blue blocks represent items that the user gives positive feedback, and gray blocks represent items that the user gives negative feedback. Fig. 1(a) shows the ideal ordering of items, Fig. 1(b) and Fig. 1(c) represent the predicting item list given by the model respectively. Fig. 1(b) has 7 pairwise errors by moving positive item to 9th. Fig. 1(b) has 6 pairwise errors by moving the two positive items to 3rd and 6th respectively. Black solid arrow and red dotted arrow represent two ways to minimize the pairwise error.

only increases NDCG value by 0.126. We find that reducing the same pairwise error has a different effect on NDCG.

Therefore, pairwise method is not the optimal method in recommendation. Because the importance of Top-N items and the items at the end of the whole item list are different. To overcome the above challenges, Lambda-based methods have been proposed by optimizing the ranking biased metrics directly such as NDCG [23]. Inspired by lambda rank, we propose a similiar lambda function as $f(D(\langle i_p, i_q\rangle|u), \zeta_u)$ where ζ_u is the current item ranking list for user u. Taking the NDCG as target, $f(D(\langle i_p, i_q\rangle|u), \zeta_u)$ is given

$$f(D(\langle i_p, i_q\rangle|u), \zeta_u) = D(\langle i_p, i_q\rangle|u)|\Delta NDCG_{p,q}| \quad (10)$$

where $\Delta NDCG_{p,q}$ represents difference of NDCG after the position of item p and item q get switched. Algorithm 2 summarizes the overall process of LambdaGAN. We take Eq. (10) as a new discriminative model for LambdaGAN, then we have the following minimax game:

$$J^{G^*,D^*} = \min_\theta \max_\phi \sum_{n=1}^{N} (\mathbb{E}_{o\sim p_{true}(\mathbf{o}|u_n)}[f(D(\mathbf{o}|u), \zeta_u)] + \mathbb{E}_{o'\sim p_\theta(\mathbf{o}'|u_n)}[1 - f(D(\mathbf{o}^{'}|u), \zeta_u)]) \quad (11)$$

However, it should be noted that the application scenarios of lambda rank is that the candidate document list is relatively small in the typical IR tasks. The candidate item list in recommendation is much larger than the candidate document list of IR tasks. For each item pair, the time to calculate $\Delta NDCG_{p,q}$ is $\mathcal{O}(T_r)$, where T_r is the time required for the current model to calculate an item rank list for the user. This time complexity is unacceptable in recommendation. On the other hand, since the number of items in the item list is very large (see the detail of datasets in next section), the magnitude of $\Delta NDCG_{p,q}$ in typical IR tasks is 10^{-1} while the magnitude turns to $10^{-4} \sim 10^{-6}$ in the most cases of recommendation. This leads to slower convergence of the algorithm and lowers the efficiency, which means the original lambda method is impractical for recommendation. Hence, following the idea of lambda rank, we redesign the lambda functions below.

C. Two Versions of LambdaGAN

Observing Fig. 1, we find that the value of NDCG will be significantly improved by moving the positive item from bottom into top of the rank list. Compared to the case of positive items in high rank, we pay more attention to the positive item in low rank. Therefore, for each item pair, we introduce two lambda function versions to replace the original $|\Delta NDCG_{p,q}|$.

1) LambdaGAN_V1: We use a reweighting term $\lambda(r(i))$ to represent the different lambda weights for each item pairs during training process. We replace $|\Delta NDCG_{p,q}|$ with Eq. (12)

$$\lambda(r(i)) = \frac{\sum_{n=0}^{r(i)} \frac{1}{n+1}}{\sum_{n=0}^{I} \frac{1}{n+1}} \quad (12)$$

$r(i)$ represents the ranking position of positive item i, I represents the whole item set. The denominator can be viewed as a normalization term for the entire formula. When r(i) is small, that is, the positive item i has a high rank in the item list. The value of $\lambda(r(i))$ will become smaller, so that this update will not cost too much loss. When r(i) is big, that is, the positive item i is at the bottom of the item list. The value of $\lambda(r(i))$ will become bigger, so that this update will push the positive items up from the bottom. Thus we have a new

discriminative model:

$$
\begin{aligned}
& f(D(\langle i_p, i_q \rangle | u), \zeta_u) \\
&= \lambda(r(i)) D(\langle i_p, i_q \rangle | u) \\
&= \frac{\sum_{n=0}^{r(i)} \frac{1}{n+1}}{\sum_{n=0}^{I} \frac{1}{n+1}} \frac{\exp(f_\phi(i_p, u) - f_\phi(i_q, u))}{1 + \exp(f_\phi(i_p, u) - f_\phi(i_q, u))}
\end{aligned} \tag{13}
$$

2) LambdaGAN_V2: In the previous section, we replace original $|\Delta NDCG_{p,q}|$ with $\lambda(r(i))$. The range of $\lambda(r(i))$ is $(\frac{1}{n+1}, 1]$. This value is less than 1, so we can think of the $\lambda(r(i))$ as a penalty for pairwise learning. When the positive item has a high rank, it will be punished less and not cost loss too much. On the contrary, when the positive item has a low rank, the value will be larger to update the gradient with a greater magnitude. From this perspective, we would like to see if lambda parameters can be used as bonus items in pairwise learning (greater than 1).

$$
\tau(r(i)) = \alpha^{\lambda(r(i))} = \alpha^{\frac{\sum_{n=0}^{r(i)} \frac{1}{n+1}}{\sum_{n=0}^{I} \frac{1}{n+1}}} \tag{14}
$$

Thus we have a new discriminative model:

$$
\begin{aligned}
& f(D(\langle i_p, i_q \rangle | u), \zeta_u) \\
&= \tau(r(i)) D(\langle i_p, i_q \rangle | u) \\
&= \alpha^{\frac{\sum_{n=0}^{r(i)} \frac{1}{n+1}}{\sum_{n=0}^{I} \frac{1}{n+1}}} \frac{\exp(f_\phi(i_p, u) - f_\phi(i_q, u))}{1 + \exp(f_\phi(i_p, u) - f_\phi(i_q, u))}
\end{aligned} \tag{15}
$$

The range of $\tau(r(i))$ is $(1, \alpha]$[1]. This value is greater than 1, we regard $\tau(r(i))$ as a bonus for pairwise learning. When the positive item ranked at the bottom of item list, $\tau(r(i))$ will assign a lager weight to the gradient, pushing the positive item to the top. Fig. 2 illustrates the value difference between lambda function version 1 and version 2. The main difference between the two versions of the lambda function is their range of values. Version 1 always acts as the penalty of pairwise learning, with the corresponding, version 2 always acts as the reward of pairwise learning.

IV. Experiments

In this section, to evaluate the effectiveness of LambdaGAN, we conduct experiments on two real world datasets.

A. Experimental Setup

1) Datasets: We use two widely used collaborative filtering datasets: Movielens-100K[2] and Netflix[3]. Consistent with the dataset processing of [6], we treat '5-star' ratings in Netflix and both '4-star' and '5-star' ratings in Movielens as positive feedback, and regard all the other items in the item list as negative feedback. In dataset partitioning, we follow the same procedures as in [17], [24]. We use the standard 5-fold cross validation. The ratio of the training set to the test set for each experiment is 4:1. The average results over 5 folds is taken as the final performance. The statistics of the preprocessed datasets are shown in Table I.

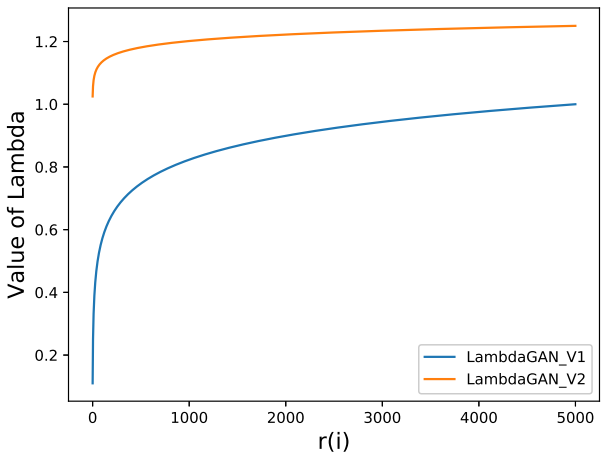


Fig. 2. The value difference between lambda function version 1 and version 2. The number of items (I) is set to 5,000 and the value of α is set to 1.25 in version 2.

TABLE I
Characteristics of the datasets.

	MovieLens-100K	Netflix
Users	943	480,189
Items	1,683	177,700
Ratings	100,000	100,480,507

2) Baseline Methods: In our experiments, we compare our model with four baseline algorithms.

$BayesianPersonalisedRanking(BPR)$. BPR [7] is a Bayesian personalized ranking learning model for implicit preference data, which is a strong baseline for Top-N recommendation.

$PairwiseRankingFactorizationMachine(PRFM)$. PRFM [3] is one of the state-of-the-art recommendation algorithm. It combines pairwise rank with factorization machine for a better performance in recommendation.

$LambdaFM$. It is a powerful Top-N recommendation baseline [2], which directly optimizes the rank biased metrics. We run the LambdaFM model with open source code[4].

$IRGAN$. This model[5] train generative model and discriminative model alternatively through adversarial processing [6].

3) Implementation Details: we pre-train G and D using training data. $L2$ regularization is set to 0.05. The learing rate is set to 1×10^{-3} on Movielens dataset, 1×10^{-4} on Netflix. Batch size is set to 32 and 128 for Movielens and Netflix respectively. The number of sampled items (K) is set to 5 and 20 for Movielens and Netflix respectively. In order

[1]The default value of α is set to 1.25, this parameter will be discussed in the following section

[2]https://grouplens.org/datasets/movielens/100k/

[3]https://www.kaggle.com/netflix-inc/netflix-prize-data

[4]https://github.com/fajieyuan/LambdaFM

[5]https://github.com/geek-ai/irgan

TABLE II
ITEM RECOMMENDATION RESULTS (MOVIELENS-100K).

	Precision@3	Precision@5	Precision@10	NDCG@3	NDCG@5	NDCG@10	MAP	MRR
BPR	0.3056	0.2912	0.2633	0.3158	0.3092	0.3009	0.1668	0.4817
PRFM	0.3290	0.3162	0.2750	0.3364	0.3291	0.3126	0.1808	0.4979
LambdaFM	0.3852	0.3561	0.3136	0.4023	0.3828	0.3650	0.2137	0.5899
IRGAN	0.3845	0.3711	0.3189	0.3959	0.3885	0.3665	0.2440	0.5836
LambdaGAN_V1	0.3940	0.3693	0.3151	0.4050	0.3893	0.3671	0.2419	0.5848
LambdaGAN_V2	**0.4189**	**0.3952**	**0.3294**	**0.4351**	**0.4203**	**0.3888**	**0.2562**	**0.6214**
Impv	8.75%	6.49%	3.29%	8.15%	8.19%	6.08%	5.00%	5.33%

TABLE III
ITEM RECOMMENDATION RESULTS (NETFLIX).

	Precision@3	Precision@5	Precision@10	NDCG@3	NDCG@5	NDCG@10	MAP	MRR
BPR	0.3271	0.3350	0.3208	0.3184	0.3264	0.3198	0.1772	0.5061
PRFM	0.3560	0.3256	0.2999	0.3765	0.3497	0.3233	0.1820	0.6001
LambdaFM	0.4017	0.3670	0.3165	0.4172	0.3890	**0.3462**	0.2031	**0.6223**
IRGAN	0.4096	0.3654	0.3154	0.4090	0.3681	0.3135	0.1712	0.5907
LambdaGAN_V1	0.3860	0.3391	0.2890	0.3932	0.3592	0.3172	0.2011	0.5887
LambdaGAN_V2	**0.4393**	**0.3911**	**0.3259**	**0.4350**	**0.3996**	0.3408	**0.2154**	0.5971
Impv	7.25%	6.57%	1.59%	4.27%	2.72%	—	6.06%	—

to ensure the fairness and comparability of the experimental results, the three algorithms (IRGAN, LambdaGAN_V1 and LambdaGAN_V2) use the same set of hyperparameters. The rest of baseline algorithms find the optimal hyperparameters by grid search.

4) Evaluation Metrics: To quantitatively evaluate our method, we adopt four rank-based evaluation metrics to measures the performance of Top-N recommendation. There are Precision@N, Normalised Discounted Cumulative Gain (NDCG@N) [17], [25], Mean Average Precision (MAP) [20] and Mean Reciprocal Ranking (MRR) [19]

B. Experimental Results

Table II and Table III demonstrate the performance of all algorithms on the two datasets. Several conclusions can be made by analyzing the experimental results. Algorithms (IRGAN, LambdaGAN_V1 and LambdaGAN_V2) based on generative adversarial nets (GAN) generally outperform traditional algorithms (BPR, PRFM and LambdaFM) in most metrics. It shows that adversarial training process can improve the performance of recommendation model. The LambdaFM algorithm using the lambda strategy also achieves good experimental results at the same time.

In Table II, LambdaGAN_V1 and LambdaGAN_V2 outperform the other baseline methods. For example, LambdaGAN_V2 gains as much as 6.49% on Precision@5 and 8.19% on NDCG@5 over strong baselines. It shows that directly optimizing the rank biased metrics with GAN loss can further enhance the performance of recommendation model.

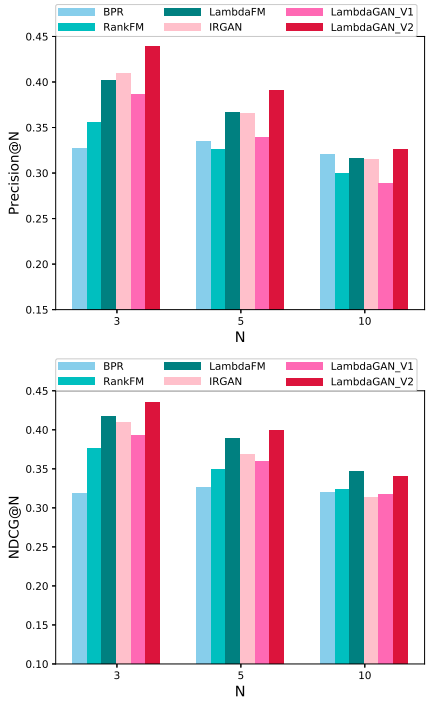


Fig. 3. Performance comparsion with respect to Precision@N, NDCG@N (Netflix).

In Table II and Table III, LambdaGAN_V2 outperforms LambdaGAN_V1 on all evaluation metrics. It indicates that lambda function performs better as being a reward than being

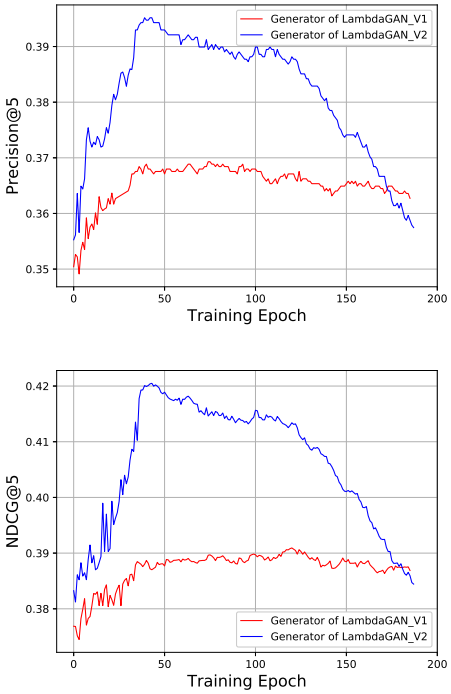
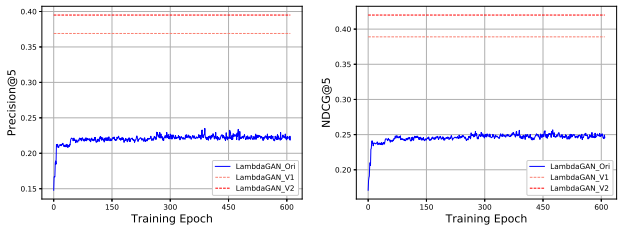


Fig. 4. Learining curves of LambdaGAN (Movielens).



(a) (b)

Fig. 5. Learining curves of LambdaGAN_Ori (Movielens).

TABLE IV
ITEM RECOMMENDATION RESULTS OF LAMBDAGAN_ORI(MOVIELENS).

	Precision@5	NDCG@5	MAP	MRR
LambdaGAN_V1	0.3693	0.3893	0.2419	0.5848
LambdaGAN_V2	0.3952	0.4203	0.2562	0.6214
LambdaGAN_Ori	0.2351	0.2561	0.1312	0.4474

a penalty in pairwise learning. In Table III, LambdaGAN_V2 significantly performs better than other methods including LambdaGAN_V1, while LambaFM performs better than LambdaGAN_V1 in Precision@N and NDCG@N. This also shows that LambdaGAN_V2 has better algorithm performance. In order to give a more intuitive display of the experimental results, we visualize the Precision@5, NDCG@5 in Fig. 3.

On the other hand, the improvement of Precision@3 (8.75%, 7.25%) is greater than Precision@5 (6.49%, 6.57%), similarly, the improvement of Precision@5 is greater than Precision@10 (3.29%, 1.59%). This law is almost identical on NDCG. The higher rating in the training set, the more likely the item is selected by LambdaGAN. It shows that the ranks of positive items have gained promotion in the recommendation list. Optimizing rank-based metrics directly can improve the model's performance.

1) Learning Curves: In order to better understand training process of adversarial learning, we visualize the learning curves of two versions of LambdaGAN as shown in Fig. 4. Due to the limited space, we only plot the precision@5 and NDCG@5. The learning curves of other metrics are similar. As can be seen from Fig. 4, LambdaGAN_V2 achieves the best performance earlier within 50 epochs. In contrast, LambdaGAN_V1 needs more than 100 epochs to reach optimal in NDCG@5. LambdaGAN_V2 performance better than LambdaGAN_V1. This is because the term of LambdaGAN_V2 is always greater than 1, which allows LambdaGAN_V2 to achieve optimality earlier and converge faster.

2) Comparison with the original lambda function: In order to compare the performance of original lambda function applied in LambdaGAN, called LambdaGAN_Ori for short, with LambdaGAN_V1 and LambdaGAN_V2. We implement the experiment with LambdaGAN_Ori on Movielens-100K. The experimental results are shown in Table IV. Moreover, Fig. 5 demonstrates the learning curve of LambdaGAN_Ori. As shown in Fig. 5, LambdaGAN_Ori still can not achieve good results after hundreds of epochs. That is because the original lambda function is originally used in the typical IR tasks, and the typical IR tasks are quite different from the Top-N recommendation tasks. In order to slove this problem, we adjust lambda function according to the characteristics of Top-N recommendation and proposed two new designed lambda functions.

3) Parameter Tuning: In LambdaGAN_V2, we propose the parameter α. Fig. 6 illustrates Precision@5 and NDCG@5 performance changes by tuning α. The model obtains best performance when α is set to 1.25. When α is set to 1, the value of lambda term is 1, lambda rank degenerates to pairwise learning. When the value of α gradually increases from 1 to 1.75, the performance of model is increased first and then decreased. We set the alpha value to 1.25 in this paper. To the best of our knowledge, the value of previous lambda functions is from 0 to 1, which can be regarded as penalties for pairwise learning. This paper proposed a new lambda function as a reward. The experiment also shows that LambdaGAN_V2 has a better effect.

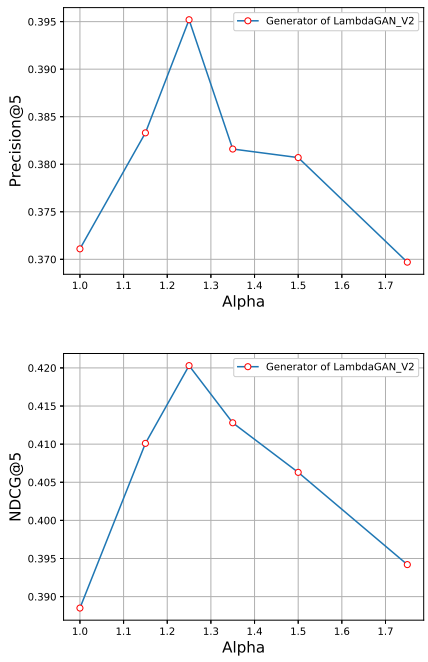


Fig. 6. Precision@5 and NDCG@5 performance under different α values (Movielens).

V. Conclusion and Future Work

This paper proposes a novel model in adversarial learning with lambda strategy for recommendation (LambdaGAN). This method combines the advantages of adversarial training and lambda rank. Besides, We modify the original lambda rank function and propose two new lambda function to meet the needs of the recommendation task. Experimental results show that LambdaGAN outperforms the other strong baselines in terms of four standard evaluation metrics.

In the future work, applying LambdaGAN to other scenarios such as web research and question answering is a direction worth studying.

VI. Acknowledgements

This research is supported by National Natural Science Foundation of China (Grant No. 61773229), Basic Scientific Research Program of Shenzhen City (Grant No. JCYJ20160331184440545), and Overseas Cooperation Research Fund of Graduate School at Shenzhen, Tsinghua University (Grant No. HW2018002).

References

[1] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *ACM Conference on Recommender Systems*, 2010, pp. 39–46.

[2] F. Yuan, G. Guo, J. M. Jose, L. Chen, H. Yu, and W. Zhang, "Lambdafm: Learning optimal ranking with factorization machines using lambda surrogates," in *ACM International on Conference on Information and Knowledge Management*, 2016, pp. 227–236.

[3] R. Qiang, F. Liang, and J. Yang, "Exploiting ranking factorization machines for microblog retrieval," in *ACM International Conference on Conference on Information & Knowledge Management*, 2013, pp. 1783–1788.

[4] S. Rendle, "Factorization machines with libfm," *Acm Transactions on Intelligent Systems & Technology*, vol. 3, no. 3, pp. 1–22, 2012.

[5] B. Schölkopf, J. Platt, and T. Hofmann, "Learning to rank with nonsmooth cost functions," pp. 193 – 200, 2006.

[6] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *SIGIR 2017 - International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.

[7] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," pp. 452–461, 2012.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.

[9] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 214–223. [Online]. Available: http://proceedings.mlr.press/v70/arjovsky17a.html

[10] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *International Conference on International Conference on Machine Learning*, 2016, pp. 1060–1069.

[11] T. Chen, Y. Liao, C. Chuang, W. T. Hsu, J. Fu, and M. Sun, "Show, adapt and tell: Adversarial training of cross-domain image captioner," *CoRR*, vol. abs/1705.00930, 2017. [Online]. Available: http://arxiv.org/abs/1705.00930

[12] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," 2016.

[13] N. Liang, H.-T. Zheng, J.-Y. Chen, A. K. Sangaiah, and C.-Z. Zhao, "Trsdl: Tag-aware recommender system based on deep learning–intelligent computing systems," *Applied Sciences*, vol. 8, no. 5, pp. 799–, 2018.

[14] X. Wang, R. Zhang, Y. Sun, and J. Qi, "Kdgan: Knowledge distillation with generative adversarial networks," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 783–794.

[15] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *International Conference on Machine Learning*, 2005, pp. 89–96.

[16] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 170–178, 1999.

[17] W. Pan and L. Chen, "Gbpr: group preference based bayesian personalized ranking for one-class collaborative filtering," in *International Joint Conference on Artificial Intelligence*, 2013, pp. 2691–2697.

[18] Z. Cao, T. Qin, T. Y. Liu, M. F. Tsai, and H. Li, "Learning to rank:from pairwise approach to listwise approach," in *International Conference on Machine Learning*, 2007, pp. 129–136.

[19] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf:learning to maximize reciprocal rank with collaborative less-is-more filtering," in *ACM Recommender Systems*, 2012, pp. 139–146.

[20] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "Tfmap:optimizing map for top-n context-aware recommendation," 2012, pp. 155–164.

[21] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[22] R. S. Sutton, "Policy gradient methods for reinforcement learning with function approximation," *Submitted to Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, 1999.

[23] C. J. C. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *International Conference on Neural Information Processing Systems*, 2006, pp. 193–200.

[24] K. Christakopoulou and A. Banerjee, "Collaborative ranking with a push at the top," pp. 205–215, 2015.

[25] B. Mcfee and G. R. G. Lanckriet, "Metric learning to rank." in *International Conference on Machine Learning*, 2010, pp. 775–782.