# Adversarial Personalized Ranking for Recommendation*

Xiangnan He
National University of Singapore
xiangnanhe@gmail.com

Zhankui He
Fudan University
zkhe15@fudan.edu.cn

Xiaoyu Du
Chengdu University of Information Technology
duxy.me@gmail.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg



## ABSTRACT

Item recommendation is a personalized ranking task. To this end, many recommender systems optimize models with pairwise ranking objectives, such as the Bayesian Personalized Ranking (BPR). Using matrix Factorization (MF) — the most widely used model in recommendation — as a demonstration, we show that optimizing it with BPR leads to a recommender model that is not robust. In particular, we find that the resultant model is highly vulnerable to adversarial perturbations on its model parameters, which implies the possibly large error in generalization.

To enhance the robustness of a recommender model and thus improve its generalization performance, we propose a new optimization framework, namely *Adversarial Personalized Ranking* (APR). In short, our APR enhances the pairwise ranking method BPR by performing adversarial training. It can be interpreted as playing a minimax game, where the minimization of the BPR objective function meanwhile defends an adversary, which adds adversarial perturbations on model parameters to maximize the BPR objective function. To illustrate how it works, we implement APR on MF by adding adversarial perturbations on the embedding vectors of users and items. Extensive experiments on three public real-world datasets demonstrate the effectiveness of APR — by optimizing MF with APR, it outperforms BPR with a relative improvement of 11.2% on average and achieves state-of-the-art performance for item recommendation. Our implementation is available at: https://github.com/hexiangnan/adversarial_personalized_ranking.

## CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Information retrieval**; **Retrieval models and ranking**;

## KEYWORDS

Personalized Ranking, Pairwise Learning, Adversarial Training, Matrix Factorization, Item Recommendation

**ACM Reference Format:**
Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommendation. In *SIGIR '18: 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8-12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3209981

## 1 INTRODUCTION

Recent advances on adversarial machine learning [30] show that many state-of-the-art classifiers are actually very fragile and vulnerable to *adversarial examples*, which are formed by applying small but intentional perturbations to input examples from the dataset. A typical example can be found in Figure 1 of [15], which demonstrates that by adding small adversarial perturbations to an image of panda, a well-trained classier misclassified the image as a gibbon with a high confidence, whereas the effect of perturbations can hardly be perceived by human. This points to an inherent limitation of training a model on static labeled data only. To address the limitation and improve model generalization, researchers then developed adversarial training methods that train a model to correctly classify the dynamically generated adversarial examples [15, 25].

While the inspiring progress of adversarial machine learning mainly concentrated on the computer vision domain where the adversarial examples can be intuitively understood, to date, there is no study about such adversarial phenomenon in the field of information retrieval (IR). Although the core task in IR is ranking, we point out that many learning to rank (L2R) methods are essentially trained by optimizing a classification function, such as the pairwise L2R method Bayesian Personalized Ranking (BPR) in recommendation [28], among others [21]. This means that it is very likely that the underlying IR models also lack robustness and are vulnerable to certain kinds of "adversarial examples". In this work, we aim to fill the research gap by exploring adversarial learning methods on item recommendation, an active and fundamental research topic in IR that concerns personalized ranking.

Nevertheless, directly grafting the way of generating adversarial examples from the image domain is infeasible, since the inputs of recommender models are mostly discrete features (*i.e.,* user ID, item ID, and other categorical variables). Clearly, it is meaningless to apply noises to discrete features, which may change their semantics. To address this issue, we consider exploring the robustness of a recommender model at a deeper level — at the level of its intrinsic

*This work is done when during the internship of Zhankui He and Xiaoyu Du at National University of Singapore. NExT research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.


*SIGIR'18, July 8–12, 2018, Ann Arbor, MI, USA*

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3209981

model parameters rather than the extrinsic inputs. Using the matrix factorization (MF) model [18, 20] trained with BPR as a demonstration (we term this instantiation as MF-BPR), we investigate its robustness to perturbations on embedding parameters. Note that MF-BPR is a highly competitive approach for item recommendation and has been used in many papers as the state-of-the-art baseline up until recently [17]. We found that MF-BPR is not robust and is vulnerable to adversarial perturbations on the parameters. This sheds light on the weakness of training with BPR, and motivates us to develop adversarial learning methods that can result in better and more robust recommender models.

As the main contribution of this work, we propose a novel *Adversarial Personalized Ranking* (APR) method to learn recommender models. With BPR as the building block, we introduce an additional objective function in APR to quantify the loss of a model under perturbations on its parameters. The formulation of APR can be seen as playing a minimax game, where the perturbations are optimized towards maximizing the BPR loss, and the model is trained to minimize both the BPR loss and the additional loss with adversarial perturbations. With a differentiable recommender model, the whole framework of APR can be optimized with the standard stochastic gradient descent. To demonstrate how it works, we derive the APR solver for MF and term the method as *Adversarial Matrix Factorization* (AMF). We conduct extensive experiments on three public datasets constructed from Yelp, Pinterest and Gowalla that represent various item recommendation scenarios. Both quantitative and qualitative analysis justify the effectiveness and rationality of adversarial training for personalized ranking. Specifically, our AMF outperforms MF-BPR with a significant improvement of 11% on average in NDCG and hit ratio. It also outperforms the recently proposed neural recommender models [17, 35] and IRGAN [31], and achieves state-of-the-art performance for item recommendation.

## 2 PRELIMINARIES

First the matrix factorization model for recommendation is described. Next the pairwise learning method Bayesian Personalized Ranking is shortly recapitulated. The novel contribution of this section is to empirically demonstrate that the MF model optimized by BPR (*a.k.a.* MF-BPR) is not robust and is vulnerable to adversarial perturbations on its parameters.

### 2.1 Matrix Factorization

MF has been recognized as the basic yet most effective model in recommendation since several years [2, 20, 40]. Being a germ of representation learning, MF represents each user and item as an embedding vector. The core idea of MF is to estimate a user's preference on an item as the inner product between their embedding vectors. Formally, let $u$ denote a user and $i$ denote an item, then the predictive model of MF is formulated as: $\hat{y}_{ui}(\Theta) = \mathbf{p}_u^T \mathbf{q}_i$, where $\mathbf{p}_u \in \mathbb{R}^K$ and $\mathbf{q}_i \in \mathbb{R}^K$ denote the embedding vector for user $u$ and item $i$, respectively, and $K$ is the size of embedding vector also called as *embedding size*. $\Theta$ denotes the model parameters of MF, which is consisted of all user embedding and item embedding vectors, *i.e.,* $\Theta = \{\{\mathbf{p}_u\}_{u \in \mathcal{U}}, \{\mathbf{q}_i\}_{i \in \mathcal{I}}\}$, where $\mathcal{U}$ and $\mathcal{I}$ denote the set of all users and items, respectively. We use $\mathbf{P}$ and $\mathbf{Q}$ to denote the embedding matrix $\mathbf{P} = \{\mathbf{p}_u\}_{u \in \mathcal{U}}, \mathbf{Q} = \{\mathbf{q}_i\}_{i \in \mathcal{I}}$ for short.

### 2.2 Bayesian Personalized Ranking

BPR is a pairwise L2R method and has been widely used to optimize recommender models towards personalized ranking [28]. Targeting at learning from implicit feedback, it assumes that observed interactions should be ranked higher than the unobserved ones. To this end, BPR maximizes the margin between an observed interaction and its unobserved counterparts. This is fundamentally different from pointwise methods [2, 17] that optimize each model prediction towards a predefined groundtruth. Formally, the objective function (to be minimized) of BPR is

$$L_{BPR}(\mathcal{D}|\Theta) = \sum_{(u,i,j) \in \mathcal{D}} -\ln \sigma(\hat{y}_{ui}(\Theta) - \hat{y}_{uj}(\Theta)) + \lambda_\Theta ||\Theta||^2, \quad (1)$$

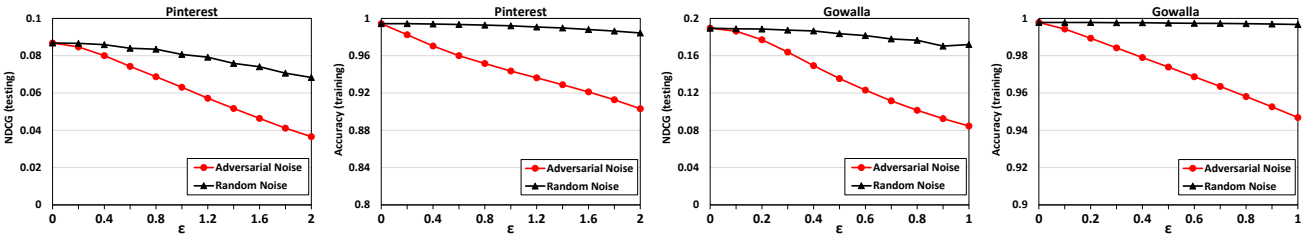
where $\sigma(\cdot)$ is the sigmoid function, $\lambda_\Theta$ are model specific regularization parameters to prevent overfitting, and $\mathcal{D}$ denotes the set of pairwise training instances $\mathcal{D} := \{(u,i,j)|i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I} \setminus \mathcal{I}_u^+\}$, where $\mathcal{I}_u^+$ denotes the set of items that $u$ has interacted with before, and $\mathcal{I}$ denotes the whole item set. Since the number of training instances in BPR is very huge, the optimization of BPR is usually done by performing stochastic gradient descent (SGD). After obtaining parameters, we can get the personalized ranked list for a user $u$ based on the value of $\hat{y}_{ui}(\Theta)$ over all items.

Owing to its rationality and ease of optimization, BPR has been used in a wide variety of scenarios [6, 7, 37–39, 41] and plays an important role in optimizing recommender models. It is worth noting that the behavior of BPR can be interpreted as a classifier — given a triplet $(u,i,j)$, it determines whether $(u,i)$ should have a higher score than $(u,j)$. Under this interpretation, a positive instance of $(u,i,j)$ means that $\hat{y}_{ui}$ should be larger than $\hat{y}_{uj}$ as much as possible to get the correct label of +1; and vice versa, a negative instance can be seen as having a label of 0.

### 2.3 MF-BPR is Vulnerable to Adversarial Noises

Inspired by the findings of adversarial examples in image classification [15, 25, 30], we are particularly interested in exploring whether the similar phenomenon exists for BPR, since it can also be seen as a classification method with triplet $(u,i,j)$ as the input. Distinct from the image domain where adding small noises to an input image shall not change its visual content, the input to BPR is discrete ID features and changing an ID feature will change the semantics of the input. For example, if we change an input $(u,i,j)$ to $(u',i,j)$ by corrupting the user ID, the semantics of the triplet becomes totally different and the label may change. As such, existing methods that generate adversarial examples for an image classifier are inappropriate for BPR.

Since it is irrational to add noises in the input layer, we instead consider exploring the robustness of BPR at a deeper level — the parameters of the underlying recommender model. It is natural to assume that a robust model should be rather insensitive to small perturbations on its parameters; that is, only when large perturbations are enforced, the model behavior should be changed dramatically. To benchmark the perturbations needed, we use random perturbations as the baseline. If we can find a way to perturb the models parameters more effectively than random perturbations, *i.e.,* resulting in a much worse recommendation performance, it




(a) Testing NDCG vs. $\epsilon$ (b) Training Accuracy vs. $\epsilon$ (c) Testing NDCG vs. $\epsilon$ (d) Training Accuracy vs. $\epsilon$

**Figure 1: Impact of applying adversarial noises and random noises to the parameters of MF-BPR on Pinterest and Gowalla.**

means that the model is not that robust and is vulnerable to certain perturbations.

**Settings**. Considering the dominant role of MF in recommendation, we choose MF as the recommender model and optimize it with BPR. We first train MF-BPR until convergence using SGD. We then compare the effect of adding random perturbations and adversarial perturbations to the embeddings of MF. For *adversarial perturbations*, we define it as the perturbations that aim to maximize the objective function of BPR:

$$\Delta_{adv} = \arg \max_{\Delta, ||\Delta|| \leq \epsilon} L_{BPR}(\mathcal{D}|\hat{\Theta} + \Delta), \tag{2}$$

where $\epsilon$ controls the magnitude of adversarial perturbations, $||\cdot||$ denotes the $L_2$ norm, and $\hat{\Theta}$ is a constant set denoting the current model parameters. As MF is a bilinear model and BPR objective function involves nonlinear operations, it is intractable to get exact maximization with respect to $\Delta$. Inspired by the fast gradient method proposed in Goodfellow *et al.* [15], we approximate the objective function by linearizing it round $\Delta$. With this approximation and the max-norm constraint, we can obtain the optimal $\Delta$ as:

$$\Delta_{adv} = \epsilon \frac{\Gamma}{||\Gamma||} \quad \text{where} \quad \Gamma = \frac{\partial L_{BPR}(\mathcal{D}|\hat{\Theta} + \Delta)}{\partial \Delta}. \tag{3}$$

As the number of training instances in $\mathcal{D}$ is huge, we sample one unobserved item $j$ to pair with an observed interaction $(u, i)$. We then perform experiments on this reduced set of examples $\mathcal{D}'$ to verify the effect of adversarial perturbations.

**Results**. Figure 1 shows the impact of applying adversarial and random perturbations to MF-BPR with different settings[1] of $\epsilon$ on our Pinterest and Gowalla datasets (details see Section 4.1). Specifically, we show the performance evaluated by NDCG@100 on the holdout testing set (Figure 1(a,c)) and the classification accuracy on the reduced training set $\mathcal{D}'$ (Figure 1(b,d)). The setting of $\epsilon = 0$ means no perturbations are used, indicating the performance of the well-trained MF-BPR. We have two main observations.

- First, both datasets show that adding adversarial noises leads to a more significant performance drop than adding random noises. For example on Gowalla, when $\epsilon$ is set to 0.4, applying random perturbations decreases the testing NDCG by 1.6%, which is a very minor impact on recommendation; in contrast, applying adversarial perturbations decreases NDCG significantly by 21.2% — 13 times larger than that of random perturbations.
- Second, even though the adversarial perturbations are derived based on partial training instances $\mathcal{D}'$ only, it has a significant adverse effect on the recommendation performance. For example on Gowalla, when $\epsilon$ is set to 1, NDCG decreases by 55.4%, whereas the training accuracy decreases by 5.1% only. Similar finding applies to the Pinterest dataset, where the drop of testing NDCG and training accuracy at $\epsilon = 2$ are 57.8% and 10.1%, respectively.

Our results indicate that MF-BPR is relatively robust to random noises, but it is rather vulnerable to certain perturbations that are purposefully designed. If a recommender model is robust and can predict user preference well, how can it be confused so much by perturbations at a small scale? The existence of such effective adversarial perturbations implies that the model learns a complicated function that overfits the training data and does not generalize well. This motivates us to develop new training methods for personalized ranking, which can lead to robust recommender models that are insensitive to such adversarial perturbations.

## 3 PROPOSED METHODS

In this section, we first present APR, an adversarial learning framework for personalized ranking. We then derive a generic solver for APR based on SGD. Lastly, we present the AMF method, an instantiation of APR that uses MF as the recommender model.

### 3.1 Adversarial Personalized Ranking

Our target is to design a new objective function such that by optimizing it, the recommender model is both suitable for personalized ranking and robust to adversarial perturbations. Due to the rationality of the BPR pairwise objective in personalized ranking, we choose it as the building block. To enhance the robustness, we enforce the model to perform well even when the adversarial perturbations (defined in Equation (2)) are presented. To achieve this, we additionally optimize the model to minimize the BPR objective function with the perturbed parameters. Formally, we define the objective function of adversarial personalized ranking as follows:

$$\begin{aligned} L_{APR}(\mathcal{D}|\Theta) &= L_{BPR}(\mathcal{D}|\Theta) + \lambda L_{BPR}(\mathcal{D}|\Theta + \Delta_{adv}), \\ \text{where} \quad \Delta_{adv} &= \arg \max_{\Delta, ||\Delta|| \leq \epsilon} L_{BPR}(\mathcal{D}|\hat{\Theta} + \Delta), \end{aligned} \tag{4}$$

where $\Delta$ denotes the perturbations on model parameters, $\epsilon \geq 0$ controls the magnitude of the perturbations, and $\hat{\Theta}$ denotes the current model parameters. In this formulation, the adversarial term $L_{BPR}(\mathcal{D}|\Theta + \Delta_{adv})$ can be seen as regularizing the model by stabilizing the classification function in BPR. As such, we also call it as *adversarial regularizer* and use $\lambda$ to control its strength. As the intermediate variable $\Delta$ maximizes the objective function to be minimized by $\Theta$, the training process of APR can be expressed as

[1]Note that we enforce the max-norm constraint of $\epsilon$ on each embedding vector in $\mathbf{P}$ and $\mathbf{Q}$, rather than the whole matrix.

playing a minimax game:

$$\Theta^*, \Delta^* = \arg\min_{\Theta} \max_{\Delta, ||\Delta|| \leq \epsilon} L_{BPR}(\mathcal{D}|\Theta) + \lambda L_{BPR}(\mathcal{D}|\Theta + \Delta), \quad (5)$$

where the learning algorithm for model parameters $\Theta$ is the minimizing player, and the procedure for getting perturbations $\Delta$ acts as the maximizing player, which aims to identify the worst-case perturbations against the current model. The two players alternately play the game until convergence. Since the focus of APR is to get a good recommender model, in practice we can determine when to stop the adversarial training by tracking how does the model perform on a validation set.

We can see that, similar to BPR, our formulation of APR leads to a general learning framework which is model independent. As long as the underlying model $\hat{y}_{ui}(\Theta)$ is differentiable, it can be learned under our APR framework using backpropagation and gradient-based optimization algorithms. There are two hyper-parameters — $\epsilon$ and $\lambda$ — to be specified in APR in addition to the ones in BPR. In what follows, we propose a generic solution for APR based on SGD.

## 3.2 A Generic SGD Solver for APR

Two optimization strategies are most widely used in recommendation — coordinate descent (CD) and stochastic gradient descent (SGD). A typical example of CD is alternating least squares [18], which iterates through model parameters and updates one parameter at a time. Note that CD is mostly used to optimize the pointwise regression loss on linear models [2]. When the optimization target involves nonlinearities, SGD becomes the default choice due to its ease in deriving the update strategy [17, 35]. Since APR involves nonlinear function in its objective function and it has a huge number of training instances (same as BPR), we optimize APR with SGD, which is easier to implement and is more efficient than CD.

The idea of SGD is to randomly draw a training instance and update model parameters with respect to the single instance only. So we consider how to optimize model parameters with respect to a randomly sampled instance $(u, i, j)$.

**Step 1. Constructing Adversarial Perturbations**. Given a training instance $(u, i, j)$, the problem of constructing adversarial perturbations $\Delta_{adv}$ can be formulated as maximizing

$$l_{adv}((u, i, j)|\Delta) = -\lambda \ln \sigma(\hat{y}_{ui}(\hat{\Theta} + \Delta) - \hat{y}_{uj}(\hat{\Theta} + \Delta)). \quad (6)$$

Here $\hat{\Theta}$ is a constant set denoting current model parameters. As such, the $L_2$ regularizer for $\Theta$ is dropped since it is irrelevant to $\Delta$. However, for many models of interest such as the bilinear MF and multi-layer neural networks [17, 35], it is difficult to get the exact optimal solution of $\Delta_{adv}$. Thus, we employ the fast gradient method proposed in Goodfellow *et al.* [15], a common choice in adversarial training [24, 26, 34]. The idea is to approximate the objective function around $\Delta$ as a linear function. To maximize the approximated linear function, we only need to move towards the gradient direction of the objective function with respect to $\Delta$, which can be derived as[2]:

$$\frac{\partial l_{adv}((u, i, j)|\Delta)}{\partial \Delta} = -\lambda(1 - \sigma(\hat{y}_{uij}(\hat{\Theta} + \Delta)))\frac{\partial \hat{y}_{uij}(\hat{\Theta} + \Delta)}{\partial \Delta}, \quad (7)$$

[2]Note the used derivative rules are: $\frac{\partial \ln x}{\partial x} = \frac{1}{x}$, and $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$.

**Algorithm 1:** SGD learning algorithm for APR.

**Input:** Training data $\mathcal{D}$, adversarial noise level $\epsilon$, adversarial regularizer $\lambda$, $L_2$ regularizer $\lambda_\Theta$, learning rate $\eta$;
**Output:** Model parameters $\Theta$;

1 Initialize $\Theta$ from BPR ;
2 **while** *Stopping criteria is not met* **do**
3  Randomly draw $(u, i, j)$ from $\mathcal{D}$ ;
  `// Constructing adversarial perturbations`
4  $\Delta_{adv} \leftarrow$ Equation (8) ;
  `// Updating model parameters`
5  $\Theta \leftarrow$ Equation (11) ;
6 **end**
7 **return** $\Theta$

where $\hat{y}_{uij}(x) = \hat{y}_{ui}(x) - \hat{y}_{uj}(x)$ for short. With the max-norm constraint $||\Delta|| \leq \epsilon$, we have the solution for $\Delta_{adv}$ as:

$$\Delta_{adv} = \epsilon \frac{\Gamma}{||\Gamma||} \quad \text{where} \quad \Gamma = \frac{\partial l_{adv}((u, i, j)|\Delta)}{\partial \Delta}. \quad (8)$$

**Step 2. Learning Model Parameters**. We now consider how to learn model parameters $\Theta$. The local objective function to minimize for a training instance $(u, i, j)$ is as follows:

$$\begin{aligned} l_{APR}((u, i, j)|\Theta) = &- \ln \sigma(\hat{y}_{ui}(\Theta) - \hat{y}_{uj}(\Theta)) + \lambda_\Theta ||\Theta||^2 \\ &- \lambda \ln \sigma(\hat{y}_{ui}(\Theta + \Delta_{adv}) - \hat{y}_{uj}(\Theta + \Delta_{adv})). \end{aligned} \quad (9)$$

In this problem, $\Delta_{adv}$ is a constant obtained from Equation (8). The derivative of the objective function with respect to $\Theta$ is as follows:

$$\begin{aligned} \frac{\partial l_{APR}((u, i, j)|\Theta)}{\partial \Theta} = &- (1 - \sigma(\hat{y}_{uij}(\Theta)))\frac{\partial \hat{y}_{uij}(\Theta)}{\partial \Theta} + 2\lambda_\Theta \Theta \\ &- \lambda(1 - \sigma(\hat{y}_{uij}(\Theta + \Delta_{adv})))\frac{\partial \hat{y}_{uij}(\Theta + \Delta_{adv})}{\partial \Theta}. \end{aligned} \quad (10)$$

Then we can obtain the SGD update rule for $\Theta$:

$$\Theta = \Theta - \eta \frac{\partial l_{APR}((u, i, j)|\Theta)}{\partial \Theta}, \quad (11)$$

where $\eta$ denotes the learning rate.

To summarize the SGD solver for APR, we give the training process in Algorithm 1. In each training step (line 3-5), we first randomly draw a instance $(u, i, j)$. We then execute the update rule for adversarial perturbations and model parameters in sequential order.

**Initialization**. It is worth mentioning that the model parameters $\Theta$ are initialized by optimizing BPR (line 1), rather than randomly initialized. This is because the adversarial perturbations are only reasonable and necessary to add when the model parameters start to overfit the data. When the model is underfitting, normal training process is sufficient to get better parameters. Besides pre-training with BPR, another feasible strategy is to dynamically adjust $\epsilon$ that controls the level of perturbations during training. For example, it is possible to learn $\epsilon$ based on a holdout validation set. We leave this exploration as future work, since we find that the current pre-training strategy with a constant $\epsilon$ works quite well.

### 3.3 Adversarial Matrix Factorization

To demonstrate how the APR works, we now provide a specific recommender solution based on MF, a basic yet very effective model in recommendation. The solution is simple and straightforward — we first train MF with BPR, and then further optimize it under our APR framework. We term the method as *Adversarial Matrix Factorization* (AMF). Figure 2 illustrates our AMF method. Since the parameters of MF are embedding vectors for users and items, we apply adversarial perturbations on the embedding vector. Given a $(u, i)$ pair, the predictive model with perturbations is defined as:

$$\hat{y}_{ui}(\Theta + \Delta) = (\mathbf{p}_u + \Delta_u)^T(\mathbf{q}_i + \Delta_i), \tag{12}$$

where $\Delta_u \in \mathbb{R}^K$ and $\Delta_i \in \mathbb{R}^K$ denote the perturbation vector for user $u$ and item $i$, respectively. Note that the max-norm constraint $||\Delta|| \leq \epsilon$ is enforced on the level of perturbation vector. To apply Algorithm 1 in AMF, we simply need to materialize Equation (8) and (11). For Equation (8), we give the key derivatives as:

$$\begin{aligned} &\frac{\partial \hat{y}_{uij}(\hat{\Theta} + \Delta)}{\partial \Delta_u} = \mathbf{q}_i + \Delta_i - \mathbf{q}_j - \Delta_j, \\ &\frac{\partial \hat{y}_{uij}(\hat{\Theta} + \Delta)}{\partial \Delta_i} = \mathbf{p}_u + \Delta_u, \quad \frac{\partial \hat{y}_{uij}(\hat{\Theta} + \Delta)}{\partial \Delta_j} = -\mathbf{p}_u - \Delta_u. \end{aligned} \tag{13}$$

To implement Equation (11), we give the key derivatives as follows:

$$\begin{aligned} &\frac{\partial \hat{y}_{uij}(\Theta)}{\partial \mathbf{p}_u} = \mathbf{q}_i - \mathbf{q}_j, \quad \frac{\partial \hat{y}_{uij}(\Theta)}{\partial \mathbf{q}_i} = \mathbf{p}_u, \quad \frac{\partial \hat{y}_{uij}(\Theta)}{\partial \mathbf{q}_j} = -\mathbf{p}_u, \\ &\frac{\partial \hat{y}_{uij}(\Theta + \Delta_{adv})}{\partial \mathbf{p}_u} = \mathbf{q}_i + \Delta_i - \mathbf{q}_j - \Delta_j, \\ &\frac{\partial \hat{y}_{uij}(\Theta + \Delta_{adv})}{\partial \mathbf{q}_i} = \mathbf{p}_u + \Delta_u, \quad \frac{\partial \hat{y}_{uij}(\Theta + \Delta_{adv})}{\partial \mathbf{q}_j} = -\mathbf{p}_u - \Delta_u. \end{aligned} \tag{14}$$

*3.3.1 Mini-batch Training for AMF.* Modern computing units such as CPU and GPU usually provide speedups for matrix-wise float operations. To leverage such speedups in learning complex models, a common strategy is to perform SGD in a mini-batch manner, *i.e.,* updating model parameters on a set of training instances rather than one instance only. In fact, many machine learning methods implemented in modern tools such as TensorFlow and Theano apply mini-batch optimizers. Since AMF plays a minimax game and has two coupled procedures, there are several ways to perform mini-batch training. Below we detail how we perform mini-batch training for AMF.

First, given the mini-batch size $S$, we randomly draw $S$ training instances from $\mathcal{D}$ and term the mini-batch as $\mathcal{D}'$. We then construct adversarial perturbations by maximizing the adversarial regularizer over the mini-batch:

$$L_{adv}(\mathcal{D}'|\Delta) = \sum_{(u,i,j)\in\mathcal{D}'} l_{adv}((u, i, j)|\Delta), \tag{15}$$

where $l_{adv}((u, i, j)|\Delta)$ has been defined in Equation (6). For each user and item[3] that occurred in $\mathcal{D}'$, we compute its perturbed vector by enforcing the max-norm constraint on $\frac{\partial L_{adv}(\mathcal{D}'|\Delta)}{\partial \Delta}$.

[3]Note that the item includes both positive item $i$ and negative item $j$. It is possible that a positive $i$ occurs in another instance as a negative item, and vice versa. This needs to be taken into account to avoid mistake.

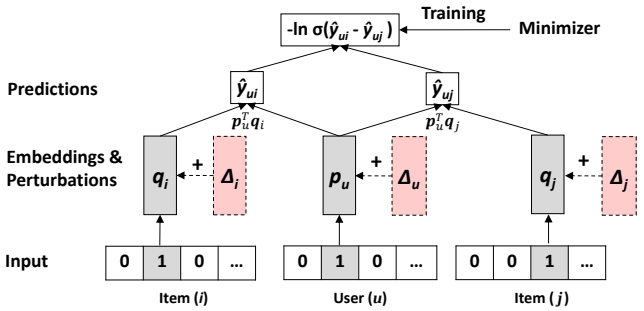



**Figure 2: Illustration of our AMF method. The perturbations $\Delta$ are enforced on each embedding vector of user and item.**

Next, we update the model parameters based on the mini-batch $\mathcal{D}'$. The APR objective function over the mini-batch is given as:

$$L_{APR}(\mathcal{D}'|\Theta) = \sum_{(u,i,j)\in\mathcal{D}'} l_{APR}((u, i, j)|\Theta), \tag{16}$$

where $l_{APR}((u, i, j)|\Theta)$ has been defined in Equation (10). Similarly, for each user and item that occurred in $\mathcal{D}'$, we perform a SGD update as $\Theta = \Theta - \eta \frac{\partial L_{APR}(\mathcal{D}'|\Theta)}{\partial \Theta}$. We iterate the above two steps until AMF reaches a convergence state or the validation performance starts to degrade.

## 4 EXPERIMENTS

As the key contribution of this work is to develop a new adversarial learning method APR for personalized ranking, we aim to answer the following research questions via experiments.

- **RQ1** How is the effect of adversarial learning? Can AMF improve over MF-BPR by performing adversarial learning?
- **RQ2** How does AMF perform compared with state-of-the-art item recommendation methods?
- **RQ3** How do the hyper-parameters $\epsilon$ and $\lambda$ affect the performance and how to choose optimal values?

Next, we first describe the experimental settings. We then report results by answering the above research questions in turn.

### 4.1 Experimental Settings

*4.1.1 Datasets.* We experiment with three publicly available datasets. Table 1 summarizes the statistics of the datasets (after all pre-processing steps). These three million-size scale datasets represent different item recommendation scenarios for business, image, and location check-in.

**Table 1: Statistics of the experimented datasets.**

| Dataset | Interaction# | Item# | User# | Sparsity |
|---|---|---|---|---|
| Yelp | 730,790 | 25,815 | 25,677 | 99.89% |
| Pinterest | 1,500,809 | 9,916 | 55,187 | 99.73% |
| Gowalla | 1,249,703 | 52,400 | 54,156 | 99.96% |

**1. Yelp**[4]. This is the Yelp Challenge data of user ratings on businesses. We use the filtered subset created by [18] for evaluating item recommendation. We find that a user may rate an item multiple times at different timestamps. Since a recommender system

[4]Downloaded from: https://github.com/hexiangnan/sigir16-eals

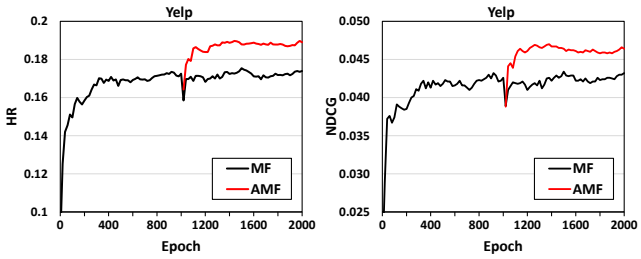



**Figure 3: Training curves of MF-BPR and AMF on Yelp.**

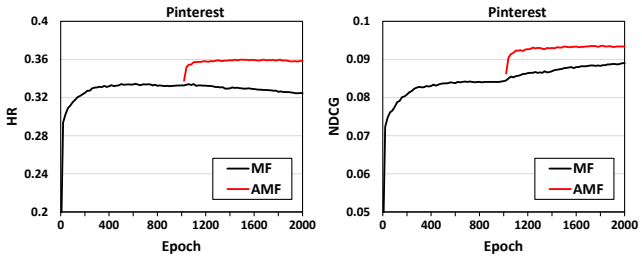



**Figure 4: Training curves of MF-BPR and AMF on Pinterest.**

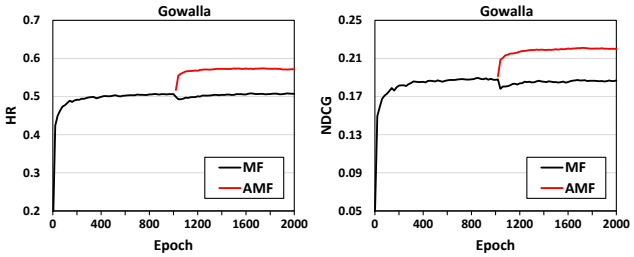



**Figure 5: Training curves of MF-BPR and AMF on Gowalla.**

typically aims to recommend items that a user did not consume before, we further merge repetitive ratings to the earliest one. This can also avoid a testing interaction appearing in the training set.

**2. Pinterest**[5]. This implicit feedback dataset was originally constructed by [13] for content-based image recommendation. We use the filtered subset created by [17] for evaluating collaborative recommendation on images. Since no repetitive interactions are found, we use the downloaded dataset as it is.

**3. Gowalla**[6]. This is the check-in dataset constructed by [23] for item recommendation. Each interaction represents a user's check-in behavior on a venue in Gowalla, a location-based social network. Same as the setting of Yelp, we merge repetitive check-ins to the earliest check-in. We then filter out items that have less than 10 interactions and users that have less than 2 interactions.

*4.1.2 Evaluation Protocol.* We employ the standard *leave-one-out* protocol, which has been widely used in item recommendation evaluation [2, 18, 28]. Specifically, for each user in Yelp and Gowalla, we hold out the latest interaction as the testing set and train a model on the remaining interactions. As the Pinterest data has no timestamp information, we randomly hold out an interaction for each user to form the testing set.

After a model is trained, we generate the personalized ranking list for a user by ranking all items that are not interacted by the user in the training set. To study the performance of top-$K$ recommendation, we truncate the ranking list at position $K$; the default setting of $K$ is 100 without special mention. We then evaluate the ranking list using *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG). HR is a recall-based metric, measuring whether the testing item is in the top-$K$ list. While NDCG is position-sensitive, which assigns higher score to hits at higher positions. For both metrics, larger values indicate better performance. We report the average score for all users, and perform one-sample paired t-test to judge the statistical significance where necessary.

*4.1.3 Baselines.* We compare with the following methods:

- **ItemPop**. This method ranks items based on their popularity, evidenced by the number of interactions in the training set. This is a non-personalized method to benchmark the performance of personalized recommendation.

- **MF-BPR** [28]. This method optimizes MF with the BPR objective function. It is a highly competitive approach for item recommendation. We tuned the learning rate and the coefficient for $L_2$ regularization.

- **CDAE** [35]. This method extends the *Denoising Auto-Encoder* for item recommendation. It has been shown to be able to generalize several latent factor models. We used the original implementation released by the authors[7], and tuned the hyperparameters in the same way as reported in their paper, including the loss function, corruption level, $L_2$ regularization and learning rate.

- **NeuMF** [17]. *Neural Matrix Factorization* is the state-of-the-art item recommendation method. It combines MF and multi-layer perceptrons (MLP) to learn the user-item interaction function. As suggested in the paper, we pre-trained the model with MF, and tuned the depth and $L_2$ regularizer for the hidden layers.

- **IRGAN** [31]. This method combines two types of models via adversarial training, a generative model that generates items for a user and a discriminative model that determines whether the instance is from real data or generated. We used the implementation released by the authors[8]. We followed the setting of the paper that pre-trains the generator with LambdaFM [38]. We tuned the learning rate and number of epochs for generator and discriminator separately, which we found to have a large impact on its performance. Further tuning of the sampling temperature did not improve the results, so we used their default settings.

This set of baselines stands for the state-of-the-art performance for the item recommendation task. In particular, CDAE and NeuMF are the recently proposed neural recommender models which have shown significant improvements over conventional shallow methods like MF and FISM [19]. IRGAN takes advantage of generative adversarial networks [14] and shows good performance on several IR tasks including recommendation in their paper.

*4.1.4 Implementation and Parameter Settings.* Our implementation is based on TensorFlow, which is available at: https://github.com/hexiangnan/adversarial_personalized_ranking. To tune the hyper-parameters, we randomly holdout one interaction for each user from the training interactions as the validation set, and we

[5]Downloaded from: https://github.com/hexiangnan/neural_collaborative_filtering
[6]Downloaded from: http://dawenl.github.io/data/gowalla_pro.zip

[7]https://github.com/jasonyaw/CDAE
[8]https://github.com/geek-ai/irgan

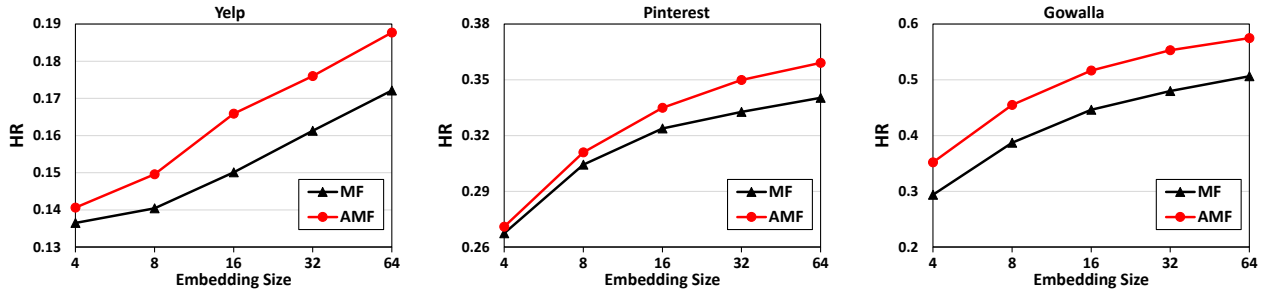



**Figure 6: Performance comparison of HR between MF-BPR and AMF with respect to different embedding sizes.**

choose the optimal hyperparameters based on NDCG@100. For a fair comparison, all models are set with an embedding size of 64 and optimized using the mini-batch Adagrad [12] with a batch size of 512; moreover, the learning rate is tuned in $[0.005, 0.01, 0.05]$. For AMF, we tune $\epsilon$ in $[0.001, 0.005, 0.01, ..., 1, 5]$ and $\lambda$ in $[0.001, 0, 01, ..., 1000]$. With MF-BPR as pre-training, AMF achieves good performance when $\epsilon = 0.5$ and $\lambda = 1$ on all datasets. As such, without special mention, we report the performance of AMF on this specific setting.

## 4.2 Effect of Adversarial Learning (RQ1)

To validate the effect of adversarial learning, we first train MF with BPR for $1,000$ epochs (mostly converged), where each epoch is defined as training the number of instances the same as the size as the training set. We then continue training MF with APR, *i.e.,* our proposed AMF method; as a comparison, we further train MF with BPR to be consistent with APR.

**1. Training Process**. Figure 3 to 5 show the performance of MF and AMF evaluated per 20 epochs on the three datasets. We can see that all figures show the same trend — after $1,000$ epochs, further training MF with APR leads to a significant improvement, whereas further training MF with BPR has little improvements. For example, on Yelp (Figure 3) the best HR and NDCG of MF-BPR are 0.1721 and 0.0420, respectively, which are improved to 0.1881 and 0.0470 by training with APR. This roughly 10% relative improvement is very remarkable in recommendation, especially considering that the underlying recommender model remains the same and we only change the way of training it.

On Gowalla (Figure 5), the improvements are even larger — 13.5% and 16.8% in terms of HR and NDCG, respectively. On Pinterest (Figure 4), we notice that HR and NDCG of MF exhibit different trends, where after $1,000$ epochs HR starts to decrease while NDCG keeps increasing. This is understandable, since HR and NDCG measure different aspects of a ranking list — NDCG is position-sensitive by assigning higher rewards to hits at higher positions while HR is not. Moreover, this points to the strength of BPR in ranking top items, owing to its pairwise objective. This observation is consistent with [18]'s finding in evaluating top-K recommendation.

**2. Improvements vs. Model Size**. Furthermore, we investigate whether the advantages of adversarial learning apply to models of different sizes. Figure 6 show the performance of MF-BPR and AMF with respect to different embedding sizes. Note that we show the results of HR only due to space limitation, and the figures of NDCG admit the same findings. First, we can see a clear trend that the performance of both methods increase with a larger embedding size. This indicates that a larger model is beneficial to top-K recommendation due to the increased modeling capability. Second, we observe that AMF demonstrates consistent improvements over MF on models of all embedding sizes. Notably, AMF with an embedding size of 32 even performs better than MF with a larger embedding size of 64 on all datasets. This further verifies the positive effect of adversarial learning in our APR method.

Lastly, it is worth noting that the improvements of AMF are less significant when the embedding size is small, compared to the setting of large embedding size. This implies that when a model is small and has limited capability, its robustness is not a serious issue. While for large models that are easy to overfit the training data, it is crucial to increase a model's robustness by learning with adversarial perturbations, which in turn can increase its generalization performance. We believe that this insight is particularly useful for the recommendation task, which typically involves a large space of input features (*e.g.,* user ID, item ID, and other attributed and contextual variables). Given such a large feature space, even a shallow embedding model like *Factorization Machine* [27] will have a large number of parameters, not to mention the more expressive deep neural networks such as *Neural Factorization Machine* [16] and *Deep Crossing* [29]. This work introduces adversarial learning to address the ranking task, providing a new means to increase the generalization ability of large models and having the potential to improve a wide range of models.

**Table 2: The impact of applying adversarial perturbations to the MF model trained by BPR and APR, respectively. The number shows the relative decrease in NDCG.**

| | $\epsilon = 0.5$ | | $\epsilon = 1.0$ | | $\epsilon = 2.0$ | |
|---|---|---|---|---|---|---|
| **Dataset** | **BPR** | **APR** | **BPR** | **APR** | **BPR** | **APR** |
| Yelp | -22.1% | -4.7% | -42.7% | -12.5% | -63.8% | -31.0% |
| Pinterest | -9.5% | -2.6% | -25.1% | -7.2% | -55.7% | -23.4% |
| Gowalla | -26.3% | -2.9% | -53.0% | -13.2% | -78.0% | -29.2% |

**3. Robustness of AMF**. We retrospect our motivating example in Section 2.3 to investigate the robustness of a model trained by APR. Table 2 shows the impact of applying adversarial perturbations to the MF model trained by BPR and APR, respectively.

We can see that by training MF with APR, the model becomes less sensitive to adversarial perturbations compared to that trained with BPR. For example, on Gowalla, adding adversarial perturbations at a noise level of 0.5 to MF-BPR decreases NDCG by 26.3%, while the number is only 2.9% for AMF. These results verify that our AMF is rather robust to adversarial perturbations, an important property that indicates good generalization ability of a model.

**Table 3: Top-$K$ recommendation performance at $K = 50$ and $K = 100$. The best result of each setting is highlighted in bold font. $*$ indicates that the improvement of the best result is statistically significant for $p < 0.01$ compared against all other methods. The last column "RI" indicates the relative improvement of AMF over the corresponding baseline on average.**

| | Yelp, HR | | Yelp, NDCG | | Pinterest, HR | | Pinterest, NDCG | | Gowalla, HR | | Gowalla, NDCG | | RI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | K=50 | K=100 | K=50 | K=100 | K=50 | K=100 | K=50 | K=100 | K=50 | K=100 | K=50 | K=100 | |
| ItemPop | 0.0405 | 0.0742 | 0.0114 | 0.0169 | 0.0294 | 0.0485 | 0.0085 | 0.0116 | 0.1183 | 0.1560 | 0.0367 | 0.0428 | +416% |
| MF-BPR | 0.1053 | 0.1721 | 0.0312 | 0.0420 | 0.2226 | 0.3403 | 0.0696 | 0.0886 | 0.4061 | 0.5072 | 0.1714 | 0.1878 | +11.2% |
| CDAE [35] | 0.1041 | 0.1733 | 0.0293 | 0.0405 | 0.2254 | 0.3495 | 0.0672 | 0.0873 | 0.4435 | 0.5483 | 0.1837 | 0.2007 | +9.5% |
| IRGAN [31] | 0.1119 | 0.1765 | **0.0361**$^*$ | **0.0465**$^*$ | 0.2254 | 0.3363 | 0.0724 | 0.0904 | 0.4157 | 0.518 | 0.1853 | 0.2019 | +5.9% |
| NeuMF [17] | 0.1135 | 0.1817 | 0.0335 | 0.0445 | 0.2342 | 0.3526 | 0.0734 | 0.0925 | 0.4558 | 0.5642 | 0.1962 | 0.2138 | +2.9% |
| AMF | **0.1176**$^*$ | **0.1885**$^*$ | 0.0350 | **0.0465**$^*$ | **0.2375**$^*$ | **0.3595**$^*$ | **0.0741**$^*$ | **0.0938**$^*$ | **0.4693**$^*$ | **0.5763**$^*$ | **0.2039**$^*$ | **0.2212**$^*$ | - |

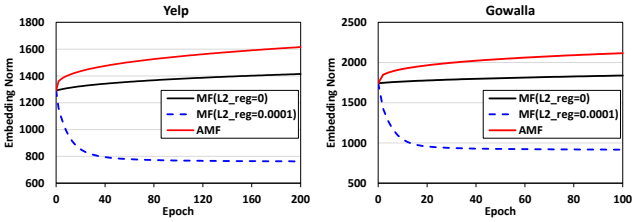



**Figure 7: The norm of embedding matrices of MF and AMF at each training epoch on Yelp and Gowalla.**

**4. Adversarial Regularization vs. $L_2$ Regularization**. The reason that APR improves over BPR is because of the adversarial regularizer. To be clear about its effect on parameter learning, we perform some micro-level analysis on model parameters. Figure 7 shows the norm of embedding matrices (*i.e.,* $||\mathbf{P}||^2 + ||\mathbf{Q}||^2$) of MF and AMF in each epoch on Yelp and Gowalla. As a comparison, we also show the effect of $L_2$ regularization, a popular technique in recommendation to prevent overfitting.

Interestingly, we find that adding adversarial regularization increases the embedding norm. This is reasonable, since we constrain the adversarial perturbations in APR to have a fixed norm, thus increasing the embedding norm is helpful to reduce the impact of perturbations. Nevertheless, simply increasing the norm by scaling up the parameters is a trivial solution, which will not improve a model's generalization performance. This provides evidence that our proposed learning algorithm indeed updates parameters in a rather meaningful way towards enhancing the model's robustness. In contrast, adding $L_2$ regularization decreases the embedding norm to combat overfitting. Based on these, we conclude that adversarial regularization improves a model's generalization in a different but more effective way from the conventional $L_2$ regularization.

## 4.3 Performance Comparison (RQ2)

We now compare our AMF with baselines. Table 3 shows the results of top-K recommendation with $K$ setting to 50 and 100. Note that we do not report results at smaller $K$, because our protocol ranks all items which makes the results at smaller $K$ exhibit large variances. More importantly, evaluating at a larger $K$ is more instructive for practitioners[9]. From Table 3, we have the following key observations:

[9]Practical recommender systems typically have two stages [33], 1) *candidate selection* that selects hundreds of items that might be of interest to a user, and 2) *ranking* that re-ranks the candidates to show top a few results. The first stage typically relies on collaborative filtering (CF) with the objective of a high recall. Thus, it is more instructive to evaluate CF with a large $K$ of hundreds, rather than a small number.

1. Our AMF achieves the best results in most cases. The only exception is on Yelp, where IRGAN outperforms AMF by a small margin in NDCG@50 and is on par with AMF in NDCG@100. For the other cases, AMF outperforms other comparing methods statistically significantly with a $p$-value of smaller than 0.01. This signifies that AMF achieves the state-of-the-art performance for item recommendation.

2. Specifically, compared to NeuMF — a recently proposed and very expressive deep learning model, AMF exhibits an average improvement of 2.9%. This is very remarkable, since AMF uses the shallow MF model that has much fewer parameters, which also implies the potential of improving conventional shallow methods with a better training algorithm.

3. Moreover, as compared to IRGAN, which also applies adversarial learning on MF but in a different way, AMF betters it by 5.9% on average. This further verifies the effectiveness of our APR method. It is worth mentioning that APR is more efficient and much easier to train than IRGAN, which needs to be carefully tuned to avoid mode collapse, while APR only requires an initialization from BPR.

4. Among the baselines, NeuMF performs the best, which verifies the advantage of nonlinear neural networks in learning the user-item interaction function. Another neural recommender model CDAE performs weaker, which only shows significant improvements over MF-BPR on the Gowalla dataset. IRGAN manages to outperform MF-BPR in most cases, which can be attributed to its improved training process, since the underlying model is also MF. Lastly, all personalized methods outperform ItemPop by a large margin, which indicates the necessity of personalization in recommendation task. This is not a new finding and has been verified by many previous works [2, 17, 28, 35, 38].

## 4.4 Hyper-parameter Studies (RQ3)

Our APR method introduces two additional hyper-parameters $\epsilon$ and $\lambda$ to control the noise level and the strength of adversarial regularizer, respectively. Here we show how do the two hyper-parameters impact the performance and also shed lights on how to set them. Due to space limitation, we show the results on the Pinterest and Gowalla datasets only, and the results on the Yelp dataset show exactly the same trend.

First, we fix $\lambda$ to the default value of 1 and vary $\epsilon$. As can be seen from Figure 8, the optimal value is around 0.5. When $\epsilon$ is too small (*e.g.,* less than 0.1), AMF behaves similarly to MF-BPR and has only minor improvements. This further verifies the positive effect of increasing the robustness of a model to perturbations on

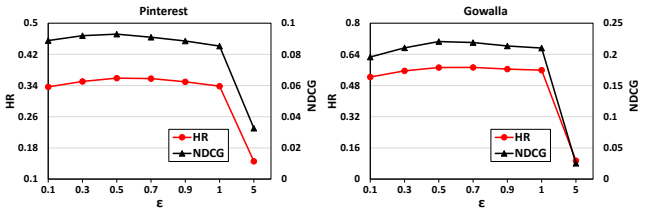



**Figure 8: Performance of AMF with respect to different values of $\epsilon$ on Pinterest and Gowalla ($\lambda$ is set to 1).**

its parameters. Moreover, when $\epsilon$ is too large (*e.g.,* larger than 1), the performance drops dramatically. This indicates that too large perturbations will destroy the learning process of model parameters. As such, our suggested setting of $\epsilon$ is 0.5 for AMF when it has been pre-trained with BPR.

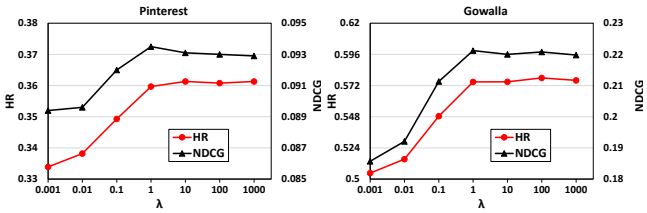



**Figure 9: Performance of AMF with respect to different values of $\lambda$ on Pinterest and Gowalla ($\epsilon$ is set to 0.5).**

Second, we fix $\epsilon$ to 0.5 and vary $\lambda$. Figure 9 shows the results. We can see that when $\lambda$ is smaller than 1, increasing $\lambda$ leads to gradual improvements. When $\lambda$ is larger than 1, further increasing it neither improves nor decreases the performance up until a large value of 1,000. This means that AMF is rather insensitive when $\lambda$ is sufficiently large to reflect the adversarial effect. As such, we suggest to set $\lambda$ to 1 (or a larger value such as 10) for AMF.

# 5 RELATED WORK

## 5.1 Item Recommendation

Due to the abundance of user feedback such ratings and purchases that can directly reflect a user's preference, research on item recommendation have mainly focused on mining the feedback data, known as collaborative filtering (CF). Among the various CF methods, matrix factorization (MF) [18], a special type of latent factor models, is a basic yet most effective recommender model. Popularized by the Netflix Challenge, early works on CF have largely focused on explicit ratings [20, 27]. These works formulated the recommendation task as a regression problem to predict the rating score. Later on, some research found that a good CF model in rating prediction may not necessarily perform well in top-K recommendation [10], and called on recommendation research to focus more on the ranking task.

Along another line, research on CF has gradually shifted from explicit ratings to one-class implicit feedback [18, 28]. Rendle *et al.* [28] first argued that item recommendation is a personalized ranking task, and such that, the optimization should be tailored for ranking rather than regression. They then proposed a pairwise learning method BPR, which optimizes a model based on the relative preference of a user over pairs of items. Later on, BPR has been used to optimize a wide range of models [6, 7, 35, 37–39, 41], being a dominant technique in recommendation. Recently, Ding *et al.* [11] improved BPR with a better negative sampler by additionally leveraging view data in E-commerce. Our proposed APR directly enhances BPR by adversarial training, having the potential to improve all existing recommender systems based on BPR.

From the perspective of models, there are many recent efforts developing non-linear neural network models for CF [1, 3, 7–9, 17, 22, 32, 35, 36, 41] to take advantage of deep learning. In particular, He *et al.* [17] argued the limitation of fixed interaction function (*i.e.,* inner product) in MF, and proposed a neural collaborative filtering (NCF) framework that learns the interaction function from data. They then designed a NCF model named NeuMF, which unifies the strength of MF and MLP in learning the interaction function. Later on, the NCF framework was extended to incorporate the neighborhoods [1] and attributes [32] of users and items, to model contexts for POI recommendation [36], to model image/video content features for multi-media recommendation [7], to model aspects in textual reviews [9], to recommend items for a group of users [5], and so on. In addition to the feed-forward NCF framework, recurrent neural networks have also been developed to handle the temporal signal in session-aware recommendation [3, 22].

## 5.2 Adversarial Learning

This work is inspired by the recent developments of adversarial machine learning techniques [15, 24–26, 30, 34]. Briefly speaking, it was found that normal supervised training process makes a classier vulnerable to adversarial examples [30], which revealed the potential issue of an unstable model in generalization. To address the issue, researchers then proposed adversarial training methods which augment the training process by dynamically generating adversarial examples [15]. Learning over these adversarial examples can be seen as a way to regularize the training process. Recently, the idea of adversarial training has been extended to learn adaptive dropout for hidden layers in deep neural networks [26].

Existing work on the emerging field of adversarial learning was largely focused on the domain of image classification. There are very few studies on adversarial learning for ranking — the core task in IR. The work that is most relevant with ours is IRGAN [31], which also employs adversarial learning, more precisely the GAN framework [14], to address the matching problem. Our APR methodology is fundamentally different from IRGAN, which aims to unify the strength of generative and discriminative models. Specifically, in the pairwise formulation of IRGAN, the generator approximates the relevance distribution to generate document (item) pairs given a query (user), and the discriminator tries to distinguish whether the document pairs are from real data or generated. Unfortunately, it is intuitively difficult to understand why IRGAN-pairwise can improve personalized ranking in item recommendation (in fact, both the original paper and their released codes only have IRGAN-pointwise for the recommendation task).

It is worth noting that in the literature of recommender systems, the concept of *robustness* usually refers to the degree that an algorithm can resist the *profile injection attack*, *i.e.,* the attack that tries to manipulate the recommendation by inserting user profiles [4]. This line of research is orthogonal to our work, since we consider

improving a recommender model by making it resistant to adversarial perturbations on its parameters. Through this way, we can get a more robust and stable predictive function, and in turn improving its generalization performance. To the best of our knowledge, this has never been explored before in the domain of IR.

## 6 CONCLUSION AND FUTURE WORK

This work contributes a new learning method for optimizing recommender models. We show that a model optimized by BPR, a dominant pairwise learning method in recommendation, is vulnerable to adversarial perturbations on its parameters. This implies the possible weakness of a model optimized with BPR in generalization. Towards the goal of learning more robust models for personalized ranking, we propose to perform adversarial training on BPR, namely, Adversarial Personalized Ranking. We develop a generic learning algorithm for APR based on SGD, and employ the algorithm to optimize MF. In our evaluation, we perform extensive analysis to demonstrate the highly positive effect of adversarial learning for personalized ranking.

In future, we plan to extend our APR method to other recommender models. First, we are interested in exploring more generic feature-based models like Neural Factorization Machines [16] and Deep Crossing [29] that can support a wide range of recommendation scenarios, such as cold-start, context-aware, session-based recommendation and so on. Second, we will employ APR on the recently developed neural CF models such as NeuMF [17] and neighbor-based NCF [1] to further advance the performance of item recommendation. The challenge here is how to properly employ adversarial training on deep hidden layers, since this work addresses the embedding layer of shallow MF model only. Lastly, it is worth mentioning that our APR represents a generic methodology to improve pairwise learning by using adversarial training. Pairwise learning is not specific to recommendation, and it has been widely applied to many other IR tasks, such as text retrieval, web search, question answering, knowledge graph completion, to name a few. We will work on extending the impact of APR to these fields beyond recommendation.

**Acknowledgments**. This work is supported by NExT, by the National Research Foundation Singapore under its AI Singapore Programme, Linksure Network Holding Pte Ltd and the Asia Big Data Association (Award No.: AISG-100E-2018-002), and by the National Natural Science Foundation of China under Grant No.: 61702300.

## REFERENCES

[1] T. Bai, J. Wen, J. Zhang, and W. X. Zhao. A neural collaborative filtering model with interaction-based neighborhood. In *CIKM*, pages 1979–1982, 2017.
[2] I. Bayer, X. He, B. Kanagal, and S. Rendle. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, pages 1341–1350, 2017.
[3] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*, pages 46–54, 2018.
[4] R. Burke, M. P. O'Mahony, and N. J. Hurley. *Robust Collaborative Recommendation*, pages 961–995. Springer US, Boston, MA, 2015.
[5] D. Cao, X. He, L. Miao, Y. An, C. Yang, and R. Hong. Attentive group recommendation. In *SIGIR*, 2018.
[6] D. Cao, L. Nie, X. He, X. Wei, S. Zhu, and T.-S. Chua. Embedding factorization models for jointly recommending items and user generated lists. In *SIGIR*, pages 585–594, 2017.
[7] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *SIGIR*, pages 335–344, 2017.
[8] X. Chen, Y. Zhang, Q. Ai, H. Xu, J. Yan, and Z. Qin. Personalized key frame recommendation. In *SIGIR*, pages 315–324, 2017.
[9] Z. Cheng, Y. Ding, X. He, L. Zhu, X. Song, and M. Kankanhalli. $A^3$NCF: An adaptive aspect attention model for rating prediction. In *IJCAI*, 2018.
[10] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.
[11] J. Ding, F. Feng, X. He, G. Yu, Y. Li, and D. Jin. An improved sampler for bayesian personalized ranking by leveraging view data. In *WWW*, pages 13–14, 2018.
[12] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
[13] X. Geng, H. Zhang, J. Bian, and T. Chua. Learning image and user features for recommendation in social networks. In *ICCV*, pages 4274–4282, 2015.
[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.
[15] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.
[16] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.
[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.
[18] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.
[19] S. Kabbur, X. Ning, and G. Karypis. Fism: Factored item similarity models for top-n recommender systems. In *KDD*, pages 659–667, 2013.
[20] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.
[21] H. Li. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2014.
[22] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428, 2017.
[23] D. Liang, L. Charlin, J. McInerney, and D. M. Blei. Modeling user exposure in recommendation. In *WWW*, pages 951–961, 2016.
[24] T. Miyato, A. M. Dai, and I. Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR*, 2017.
[25] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *CVPR*, pages 86–94, 2017.
[26] S. Park, J.-K. Park, S.-J. Shin, and I.-C. Moon. Adversarial dropout for supervised and semi-supervised learning. In *AAAI*, 2018.
[27] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.
[28] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.
[29] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *KDD*, pages 255–262, 2016.
[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.
[31] 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*, pages 515–524, 2017.
[32] X. Wang, X. He, L. Nie, and T.-S. Chua. Item silk road: Recommending items from information domains to social users. In *SIGIR*, pages 185–194, 2017.
[33] Z. Wang, Z. Jiang, Z. Ren, J. Tang, and D. Yin. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *WSDM*, pages 619–627, 2018.
[34] Y. Wu, D. Bamman, and S. Russell. Adversarial training for relation extraction. In *ACL*, pages 1778–1783, 2017.
[35] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, pages 153–162, 2016.
[36] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han. Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *KDD*, pages 1245–1254, 2017.
[37] W. Yu, H. Zhang, X. He, X. Chen, L. Xiong, and Z. Qin. Aesthetic-based clothing recommendation. In *WWW*, pages 649–658, 2018.
[38] 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 *CIKM*, pages 227–236, 2016.
[39] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362, 2016.
[40] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua. Discrete collaborative filtering. In *SIGIR*, pages 325–334, 2016.
[41] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft. Joint representation learning for top-n recommendation with heterogeneous information sources. In *CIKM*, pages 1449–1458, 2017.