Adversary or Friend? An adversarial Approach to Improving Recommender Systems

Pannaga Shivaswamy
pshivaswamy@netflix.com
Netflix Inc.
Los Gatos, CA, USA

Dario Garcia-Garcia*
dario.garcia@feverup.com
Fever Inc.
New York, NY, USA

ABSTRACT

Typical recommender systems models are trained to have good average performance across all users or items. In practice, this results in model performance that is good for some users but sub-optimal for many users. In this work, we consider adversarially trained machine learning models and extend them to recommender systems problems. The adversarial models are trained with no additional demographic or other information than already available to the learning algorithm. We show that adversarially reweighted learning models give more emphasis to dense areas of the feature-space that incur high loss during training. We show that a straightforward adversarial model adapted to recommender systems can fail to perform well and that a carefully designed adversarial model can perform much better. The proposed models are trained using a standard gradient descent/ascent approach that can be easily adapted to many recommender problems. We compare our results with an inverse propensity weighting based baseline that also works well in practice. We delve deep into the underlying experimental results and show that, for the users who are under-served by the baseline model, the adversarial models can achieve significantly better results.

Baseline ↗

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Machine learning**.

KEYWORDS

Recommender Systems, Adversarial Learning, Fairness, EASE

ACM Reference Format:
Pannaga Shivaswamy and Dario Garcia-Garcia. 2022. Adversary or Friend? An adversarial Approach to Improving Recommender Systems. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3523227.3546784

1 INTRODUCTION

Recommender systems are becoming ubiquitous with many of our online interactions on the web and via the phone relying on them. These systems are used in web-search, product recommendation [30], video [4] and movie [9] recommendation, feed ranking [2] etc. Given their ubiquitous nature, it is important to understand how they perform for different users and modify them in ways that nudge them to work better for users whom they may be currently under-serving. In this paper, we propose new adversarial training methodologies for recommender systems and show that they result in systems that work better for users that may be under-served. → Being able to selectively improve for users is the goal.

Many existing recommender systems are trained using machine learning approaches using a variety of techniques [14–16, 19, 22, 24, 25, 33]. Most of these approaches rely on training a model that minimizes a given loss on the set of observations that are available at the time of training. The final goal for many recommender systems is to optimize some utility function representing a businesses goal. → Common goal. In most cases, the underlying models are optimized for average utility across the user base (and item base). A result of this is that the performance of the recommender system can be very skewed across users (and/or items). This skew can come from a variety of reasons. For example, a model can focus on majority behaviors at the expense of some minority tastes; or similarly ignore users with fewer or very diverse existing interactions in the training set which makes them harder to make predictions for. → Issues The question then arises: how can we optimize a recommender system such that we not only achieve higher overall average performance but also reduce the likelihood of under-served pockets of users? Addressing such questions would not only enhance our understanding of the limitations of the existing approaches, but will also pave the way towards making them better for under-served populations. The title of this paper is motivated from the fact that although the models we propose in this paper might be trained in an adversarial fashion, they behave benignly in the sense that they improve the performance for many under-served users. → Goal

Recently, there is considerable interest in the literature studying machine learning problems from the fairness angle. Many of these approaches are aimed at improving the performance over all demographic groups [12, 18] even when the demographics are unknown at an individual user level. This paper is motivated from such approaches in that we also do not rely on having any demographic information.

The following are the major contributions of this work. First, we consider variants of adversarial supervised learning models proposed in [18] and give an interpretation of those models. This interpretation motivates us to extent adversarial training approaches to recommender models. We propose generic adversarial recommender problems that are applicable for many of the existing recommender system training algorithms and then specifically consider the recently proposed EASE model [25] and build adversarial

EASE model used to compare.
↳ Auto-encoder based

*This work was done while at Netflix.

RecSys '22, September 18–23, 2022, Seattle, WA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9278-5/22/09.
https://doi.org/10.1145/3523227.3546784

models on top of the EASE model. We propose a straightforward adversarial model (based on [18]) and then extend it to a rank-loss based adversarial model. The experiments section show that the straightforward extension does not work well. The rank-loss based adversarial model works better empirically compared to an adversarial model. In this work, we see that there is a large spectrum of models that achieve similar or even significantly better overall performance compared to EASE, while, surprisingly, also improving the performance vastly for under-served users compared to the baseline models.

The rest of this paper is organized as follows. In Section 2 we review the related literature. In Section 3 we start with the basic supervised problem, consider adversarial variants of the learning problem and provide interpretation of what they achieve. In Section 4 we extend the adversarial models to the recommender systems problem and study some extensions. We present experiments in Section 5 and finally conclude in Section 6.

2 RELATED WORK

There is abundant literature on recommender systems models covering a wide spectrum of techniques. Some of the example techniques range from matrix factorization [16], SLIM model and its variants [22], many recent deep learning models [14, 15, 24, 33] and variational inference [19] etc. Recently an Auto-Encoder model (called the EASE model [25, 26]) has been shown to achieve very impressive performance while exhibiting a closed form solution. Although much of the work in this paper is generic in nature and applicable to many recommender systems, as a use-case we will work with the objective from the EASE model.

There is significant interest in the literature on adversarial machine learning in the recent years. This includes, Generative Adversarial Networks (or GANS) [5, 10], to robust machine learning of different kinds [8, 12, 27, 31]. There is also recent interest in applying adversarial machine learning from a fairness perspective. Many of these approaches [7, 12, 18] are for achieving some notion of fairness across often unknown groups. In contrast, the focus of the work in this paper is to see if adversarially trained recommender systems improve the performance on under-served users. In our work there are no groups being defined a-priori explicitly or implicitly via e.g. demographic characteristics. The subsets of users which we want to "uplift" emerge naturally from the performance of the baseline model. There is also literature looking at the problem of presence of adversarial training examples in recommender systems [6] looking at it from an adversarial regularization viewpoint.

Inverse propensity weighting (IPW [20]) is a classic technique that can be used for overcoming biases in the data. For example, if we know some users who do not have much data are under-served, they could be up-weighted using IPW techniques such that a model spends more capacity to improve their performance. Such techniques can be used even in the presence of clean validation data [28] using more sophisticated approaches. Many approaches under the name of domain adaptation, transfer learning, covariate shift corrections [3, 32] are also considered in the literature to build models that work well for specific segments. However, the goal in this paper is not to build specific models for specific groups but to build a single model that has a different trade-off characteristic compared to users that are under-served by a model.

The most closely related work to this paper comes from [18]. The main focus of [18] is to achieve fairness without demographics in the general supervised learning problem. Our main formulations are motivated from the approaches in [18]. However, our work applies to recommender system problems which results in different models for user level adversarial models. Moreover, we also give additional insights into the formulation itself which makes the idea of "computational identifiable" from [18] a little more intuitive. In this work, we also apply the adversarial ML models on large scale recommender problems compared to much smaller datasets presented in [18]. In [13], adversarial approaches are proposed for recommender problems where they consider perturbations around a model rather than adversarially reweighing at a user level which is the main focus of this paper. Finally, Distributionally Robust Optimizations [29] have been considered for improving worst-case user experience in recommender problem. Their approach relies on apriori identifying groups for the learner to focus on which is in contrast to the work in this paper which doesn't rely on prior identification of groups.

→ works for large scale recsys models as well.

↳ No assumptions on user groups apriori in this paper.

3 ADVERSARIAL MODELS IN STANDARD SUPERVISED LEARNING

We first consider the standard supervised learning models. Here we assume that n training examples $(x_i, y_i)_{i=1}^n \in \mathcal{X} \times \mathcal{Y}$ are available during training. We assume that $l(x_i, y_i; w)$ denotes a loss function parameterized by the model parameters w. In standard machine learning the empirical loss over the training examples is minimized along with a regularization term $C > 0$ on the model parameters. That is, we typically minimize,

$$\boxed{\min_w \frac{1}{n} \sum_{i=1}^{n} l(x_i, y_i; w) + \frac{C}{2} \|w\|^2.} \tag{1}$$

Adversarial machine learning models have been proposed [18] motivated from the Rawlsian fairness criterion [23], which posits maximizing some utility over the worst case group. By improving the performance of the worst performing group, it is ensured that no particular group has a poor performance. The Rawlsian min-max criterion assumes that we know the exact group memberships and thus we can consider the utility for the worst case group. However, in many learning and recommender systems problems, we do not have access to such group information at the individual user level. In such cases, the learning objective is modified such that the loss is minimized with a weighted score from an adversary that is also learned along with the model. Rather than minimizing the loss over different groups, each individual example is endowed with an adversary score and we consider the following optimization problem which is very similar to the one proposed in [18]:

$$\boxed{\min_w \max_a \sum_{i=1}^{n} \sigma(a^\top \phi(x_i, y_i)) l(x_i, y_i; w)/z + \frac{C}{2}\|w\|^2 - \frac{B}{2}\|a\|^2.} \tag{2}$$

Compared to the empirical risk above in formulation (1) above, in the formulation (2), we still have a minimization over w which represents the learner. However, in addition to the learner, now we

→ Normalize adversary weights so they sum to one over all examples.

L_2 regularizer for adversary model

also have an adversary which is trying to maximize the objective over $a \in \mathbf{R}^m$ via the scores that are going through the sigmoid function ($\sigma(\theta) = \frac{1}{1+\exp(-\theta)}$) which is always between zero and one. We consider a joint feature map $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}^m$ as a feature representation used for the adversary. Note that the adversarial model can be dependent on both x and y. Allowing adversary model to put arbitrary weights on the examples can be detrimental to the learning problem. Given complete flexibility, the adversary can pick a model such that it is impossible for the learner to do well on the learning problem. We thus constrain the adversary in two ways. We add an ℓ_2 regularizer term with weight $0.5B$ on the adversary model parameters. We also normalize the adversary weights such that they sum to one over all the examples. This is achieved by means of the normalizer z which is defined as, $z = \sum_{i=1}^{n} \sigma(a^\top \phi(x_i, y_i))$. We also do not have $1/n$ in formulation 2 since the normalizer z is already such that the weights over all the examples sum to one.

It is clear that performing the inner maximization over adversarial model a increases the adversarial weights where the losses are generally higher. However, since the weights on the examples are coming from a model, they have some smoothness property in the sense that examples in similar neighborhoods are likely to get similar weights. This property was termed "computationally identifiability" in [18]. Here, we try to understand the adversary a little bit better with the following analysis.

The full problem in (2) involves both a minimization over w and a maximization over a. However, we can look at the adversary optimization of a for a fixed w to see how the adversary distributes it weights over the training examples. Starting with the maximization problem over a at a particular w where we denote $l(x_i, y_i; w)$ by l_i, we have,

$\to$ At a particular w

$$\max_a \sum_{i=1}^{n} \sigma(a^\top \phi(x_i, y_i)) l_i / z - \frac{B}{2} \|a\|^2. \quad (3)$$

The gradient of the above objective with respect to a is given by,

$$\frac{z \sum_{i=1}^{n} \sigma'(a^\top \phi(x_i, y_i)) \phi(x_i, y_i) l_i}{z^2}$$
$$- \frac{\sum_{i=1}^{n} \sigma(a^\top \phi(x_i, y_i)) l_i \sum_{j=1}^{n} \sigma'(a^\top \phi(x_j, y_j)) \phi(x_j, y_j)}{z^2} - Ba$$
$$= \sum_{i=1}^{n} p_i (1 - \sigma(a^\top \phi(x_i, y_i))) l_i \phi(x_i, y_i)$$
$$- \sum_{i=1}^{n} p_i l_i \sum_{j=1}^{n} p_j (1 - \sigma(a^\top \phi(x_j, y_j))) \phi(x_j, y_j) - Ba$$
$$= \sum_{i=1}^{n} p_i (1 - \sigma(a^\top \phi(x_i, y_i)))(l_i - \mathbf{E}_p[l]) \phi(x_i, y_i) - Ba$$

$\to$ Average of l_is

In the above equations, $\sigma(a^\top \phi(x_i, y_i))/z$ is denoted by p_i which are the normalized weights and note that $\sum_{i=1}^{n} p_i = 1$ and $0 \le p_i \le 1$. On equation two, we used the fact that $\sigma'(\theta)$ is the derivative of the sigmoid function and is given by $\sigma(\theta)(1 - \sigma(\theta))$. We denoted the average of the losses l_i's induced by the distribution p_i as $\mathbf{E}_p l$ in the final equation.

We next look at how the adversary solution looks at any local optima. We do this by equating the above gradient to zero. Because of the choice made for the adversary a with the sigmoid, there is no closed form solution to this problem, but we can still get a better understanding of the factors that play a role in high and low adversary scores by looking at the following equation. Setting the above derivative to zero, we have that, at any local maximum over the adversary,

$$\boxed{a = \frac{1}{B} \sum_{i=1}^{n} p_i (1 - \sigma(a^\top \phi(x_i, y_i)))(l_i - \mathbf{E}_p[l]) \phi(x_i, y_i).}$$

Next, we consider the prediction from the adversarial model (unnormalized and without the sigmoid) on a generic feature-map $\phi(x_j, y_j)$ at any local maximum, which is given by the following:

$$\boxed{\begin{aligned} & a^\top \phi(x_j, y_j) \\ = & \frac{1}{B} \sum_{i=1}^{n} \frac{\sigma'(a^\top \phi(x_i, y_i))}{z} (l_i - \mathbf{E}_p[l]) \phi(x_i, y_i)^\top \phi(x_j, y_j). \quad (4) \end{aligned}}$$

In the above equation we wrote $\sigma'(a^\top \phi(x_i, y_i))$ for $\sigma(a^\top \phi(x_i, y_i))(1 - \sigma(a^\top \phi(x_i, y_i)))$ to fit the equation on the line. The above equation describes a prediction for a given example in terms of predictions over all the examples, losses and the inner product between examples (which can be interpreted as a similarity between $\phi(x_i, y_i)$ and $\phi(x_j, y_j)$). The weights given by $\frac{\sigma'(a^\top \phi(x_i, y_i))}{z}$ in the equation (4) can be interpreted as the (local) optimal weights over the examples at a particular local optima (a).

We look at the equation (4) as "explaining" a score in terms of the factors involved. Every single examples contributes towards the adversary score depending on its similarity to other examples, the loss at the point and a factor that is determined by the adversarial score itself. Suppose there is an example $\phi(x_j, y_j)$ where the adversary gives a high (positive) score, i.e., $a^\top \phi(x_j, y_j) >> 0$. What does it imply from the above equation? First, we note that if $a^\top \phi(x_j, y_j)$ is high, $\sigma'(a^\top \phi(x_j, y_j))$ is small. Thus, the contribution from the example itself is low (unless the loss the loss at that point l_j is extremely high). If we now consider the contributions from all the other examples, if there are many other examples that are similar to $\phi(x_j, y_j)$, that is, with high inner product (i.e. very similar examples) and their losses are higher than the mean loss (i.e., $l_i - \mathbf{E}_p[l] \ge 0$) then they would contribute towards the high score. Thus, there is an interplay between the loss value and similarity of an example to other examples that both play a role in the final adversary scores. We can similarly infer that examples with low loss values without many other neighbors with high loss values would end up getting a low adversary score.

Factors affecting the score

A cursory look at the optimization problem (2) might make it seem like only the loss values determine the adversary scores. However, the analysis above shows that it is much more nuanced. We saw that a high adversary score implies either an extremely high loss or many similar training examples also with high losses. This interpretation matches with the intuition provided in [18] that adversarial reweighting models give high weights to what they refer to as "computationally identifiable regions". Thus, learning in an adversarial framework can improve the model performance in dense regions of high losses. With this motivation, in the rest of this paper we focus on extending the adversarial learning framework from the formulation (2) to the recommender systems problem such that we improve the model on regions of high density where a model might be currently performing poorly. In the next section, we propose adversarial recommender models motivated by the analysis in this section and the approach in equation (2).

4 ADVERSARIAL RECOMMENDER MODELS

Now we consider a recommender systems problem. We assume that there are m users and n items and that a matrix of binary observations[1] $X \in \{0,1\}^{m\times n}$ is available for training. These binary observations could be clicks from users on items or views from users on movies or TV/shows. The end-goal in recommender systems problems is to rank items to users such that the ranking is meaningful to its users. We propose adversarial models for the recommendation problem in this section. A typical way to train a recommender system model is by minimizing a loss over the entire training set where the loss is a measure of the difference between the true label and the predicted label. Following represents a generic way to train a recommender systems model:

$$\boxed{\min_W \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} l(X_{ij},[f(X;W)]_{ij}) + \frac{C}{2}\Omega(W)} \quad (5)$$

In the above equation, $W \in \mathcal{W}$ represents a generic set of parameters learned such that $[f(X;W)]_{ij}$ denotes the prediction for the i^{th} user and j^{th} item. We denote by $l : \mathbf{R}\times\mathbf{R} \longrightarrow \mathbf{R}^+$ a loss function between an observation X_{ij} and a prediction $[f(X;W)]_{ij}$ for the i^{th} user and j^{th} item. Ω represents a generic regularization term (such an explicit ℓ_1 or ℓ_2 regularization). For particular choices of W, f and Ω we get different ways of training a recommender systems model.

The same way we went from a supervised learning problem minimizing the empirical loss in Eqn. (1) to an adversarially trained model (2), we propose the following generic adversarial model for the recommender systems problem. In the following the parameters $A \in \mathcal{A}$ denote the parameters over the adversarial model.

$$\boxed{\begin{aligned}\min_W \max_A \sum_{i=1}^{m}\sum_{j=1}^{n} l(X_{ij},[f(X;W)]_{ij})\underbrace{\sigma([g(X;A)]_{ij})}_{\text{Adversarial model predictions}}/z \\ + \frac{C}{2}\Omega(W) - \frac{B}{2}\Omega(A)\end{aligned}}$$

In the above equation, the prediction from the adversarial model on the i^{th} user and j^{th} item is denoted by $[g(X;A)]_{ij}$. In the above formulation, z denotes a normalization factor such that the adversarial scores do not change the overall weight between the loss term and the regularization terms. We normalize z such that over all i and j the weights add to one, that, is, $z = \sum_{i=1}^{m}\sum_{j=1}^{n}\sigma([g(X;A)]_{ij})$. The adversary predicts a score for every (user, item) pair in the above formulation.

While the above formulation proposes an adversarial model that gives a score for every user and item, in this paper, we only consider adversarial models that reweight users rather than users and items and study the resulting formulations in depth. The following formulation specializes the above formulation such that the adversary

This paper only aims to reweight users

[1]This assumption of binary observations is to keep the presentation simple, the framework itself is easily extended to non-binary observations as well.

weights are only at the user level:

$$\boxed{\begin{aligned}\min_W \max_A L(W,A) = \sum_{i=1}^{m}\sum_{j=1}^{n} l(X_{ij},[f(X;W)]_{ij})\sigma([g(X;A)]_i)/z \\ + \frac{C}{2}\Omega(W) - \frac{B}{2}\Omega(A)\end{aligned}} \quad (6)$$

The adversarial model predictions $g(X,A)$ are assumed to give a score for every user and we weigh the i^{th} user with their adversarial score. The normalization factor z is a summation over the user level scores, .i.e.,

$$\boxed{z = \sum_{i=1}^{m}\sum_{j=1}^{n}\sigma([g(X;A)]_i) = n\sum_{i=1}^{m}\sigma([g(X;A)]_i).}$$

Note that the models presented so far are very generic and they can be applied to a wide range of recommender problems such as SLIM [22], Matrix Factorization [16] etc. with minor modifications.

4.1 Optimizing adversarial models

Solving the adversarial learning formulation in (6) involves a minimization over the learner parameters W and a maximization over the adversary parameters A. We thus solve such models using gradient descent/ascent. We consider all columns of X and a randomly sampled subset of rows (corresponding to users) and updated W and A in each iteration. We considered the algorithm outlined in Algorithm 1 to solve the min-max problem by first taking a few steps in the learner alone (in the first K epochs) and then updating both the learner and adversary each step. This optimization follows steps similar to those in [18]. More details about the specific instance of the algorithm used in our experiments can be found along with our experiments in Section 5. So far, we considered generic formulations for the recommender problems. Next, we specialize the above formulation to the EASE model which is the primary model used in the experiments section of this paper. We first give a brief background on EASE below for completeness of presentation.

Algorithm 1 Algorithm to Train Adversarial Recommender Models (ARM)

```
1: Input: Mini-batch Size $l$, Regularization Parameters $C > 0, B > 0$, Number of epochs $N$, warm-up epoch $K$
2: for $epoch \leftarrow 1$ to $N$ do
3:     Get a mini-batch from $X$ with $l$ rows and all columns
4:     if $epoch > K$ then
5:         Compute mini-bath gradient over $A : \nabla_A L(W,A)$ (loss defined in eqn. (6))
6:         Make a solver update (gradient ascent) on $A$
7:     end if
8:     Compute mini-bath gradient over $W : \nabla_W L(W,A)$
9:     Make a solver update (gradient descent) on $W$
10: end for
```

4.2 The EASE Model

The EASE [25] model is an auto-encoder model which has been shown to be highly effective on recommendation problems. The goal in EASE is to predict the rating of an item for a user based on the ratings from all the other items for the same user in the observation matrix. It uses the squared loss to measure the difference between the predicted values and the actual values and a Frobenius norm regularization on the model parameters. The EASE model objective is stated below:

$$\boxed{\begin{aligned}&\min_{W} \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(X_{ij} - [XW]_{ij})^2 + \frac{C}{2}\|W\|_F^2 \\ &\text{s.t. } \operatorname{diag}(W) = 0.\end{aligned}} \tag{7}$$

Frobenius Norm

In the above formulation $W \in \mathbf{R}^{n\times n}$ is the item-item predictive matrix that we are learning. $[XW]_{ij}$ denotes the prediction for the i^{th} user and the j^{th} item, that is the i^{th} row and j^{th} column of the matrix XW. The diagonal constraint on W ensures that we do not predict an item score from the observation itself. This model is also similar to the SLIM model [22] that has been extensively studied in the literature, except that there is no non-negativity constraint on the weights learned. It was shown [25] that the above optimization has a closed form solution and that the resulting model achieves state-of-the-art performance. Note that the formulation above is exactly the same as that proposed in [25] except that we divide the first term by the number of elements in the matrix to be consistent with the rest of the presentation in this paper. This makes it same as the one in [25] up to a scaling factor on the regularization parameter C.

One of the baselines we consider in this paper is a so-called Inverse Propensity Weighting (IPW) model [20] at the user level. Suppose we have a weight $\theta_i > 0$ for user i, the weighted EASE objective in this case becomes,

EASE with IPW

$$\boxed{\begin{aligned}&\min_{W} \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\theta_i(X_{ij} - [XW]_{ij})^2 + \frac{C}{2}\|W\|_F^2 \\ &\text{s.t. } \operatorname{diag}(W) = 0.\end{aligned}} \tag{8}$$

Moving θ_i inside the objective, we get the inner term in the objective as $(\sqrt{\theta_i}X_{ij} - \sqrt{\theta_i}[XW]_{ij})^2$. We can thus use the same closed form solution for EASE for obtaining EASE - IPW soluton simply by multiplying the i^{th} row of X by $\sqrt{\theta_i}$. More details of the IPW method, the weights used etc. can be found in the experiments section.

4.3 User Level Adversarial Recommender Model (ARM) from EASE

Based on the generic user level formulation in (6), a straight-forward extension of the EASE model is as follows. We denote the adversary model by a vector $A \in \mathbf{R}^n$ The adversary simply gives a score of $[XA]_i$ for the i^{th} user. Or equivalently, denoting the vector for the i^{th} user by $X_i \in \mathbf{R}^{1\times n}$, the score given is $x_i a$ which is then passed through the sigmoid and normalized by z:

$$\boxed{\begin{aligned}&\min_{W}\max_{A} \sum_{i=1}^{m}\sum_{j=1}^{n}(X_{ij} - [XW]_{ij})^2\sigma(X_iA)/z + \frac{C}{2}\|W\|_F^2 - \frac{B}{2}\|A\|^2 \\ &\text{s.t. } \operatorname{diag}(W) = 0\end{aligned}} \tag{9}$$

Multi-hot encoding of user

The above formulation takes the multi-hot encoding vector of a user (depending on which of the movies they have watched) and simply takes the inner product with the vector a effectively selecting the elements of a corresponding to watched movies then adds those elements. This score is passed through the sigmoid and normalized to get the adversary score.

Algorithm 2 Algorithm to Train Adversarial Models with Rank Loss (R-LARM)

```
1:  Input: Mini-batch Size $l$, Regularization Parameters $C > 0, B >$
    $0$, Number of epochs $N$, warm-up epoch $K$
2:  for $epoch \longleftarrow 1$ to $N$ do
3:      Get a mini-batch from $X$ with $l$ rows and all columns
4:      if $epoch > K$ then
5:          Compute mini-bath gradient over $A : \nabla_A P(W, A)$ (loss
    defined in eqn. (10))
6:          Make a solver update (gradient ascent) on $A$
7:      end if
8:      Compute mini-bath gradient over $W : \nabla_W L(W, A)$
9:      Make a solver update (gradient descent) on $W$
10: end for
```

4.4 Rank-Loss Based Adversarial Recommender Model (R-LARM)

We note that in Algorithm 1, the adversary gradients are computed in each iteration based on the loss $L(W, A)$. Rather than working with the same $L(W, A)$ objective for the learner and the adversary, we consider a variant where for the adversary step we considered a loss $P(W, A)$ that is better in line with the ultimate objective that we care about. In ranking problems, for example, we are interested in metrics such as the NDCG, Recall or MRR. So, rather than taking gradient ascent steps with the gradients $(\nabla_A L(W, A))$ which is based on the same loss as on Line 8, we take gradients with a more appropriate (to the task at hand) rank-loss $P(W, A)$ on the adversary. We define a loss based on NDCG, as $1 - NDCG(k, W)_i$ where $NDCG(k, W)_i$ is the NDCG with the model W for the user i, we have the following rank loss for the adversary:

$$\boxed{\max_{A} P(W, A) := \sum_{i=1}^{n}(1 - NDCG_i(k, W))\sigma([XA]_i)/z - \frac{B}{2}\|A\|^2} \tag{10}$$

Here NDCG is defined in the standard way at k using the model W. First we compute the DCG(k) by taking the discounted relevance (0 or 1 in our case) of the items ranked using the model W and summing over it over positions from 1 to K. That, is,

$$\boxed{DCG(k, W) = \sum_{s=1}^{k}\frac{rel_s^W}{\log_2(s+1)}} \tag{11}$$

where rel_s^W denotes the binary relevance of the item at position s when ranked by the scores derived from the model W. NDCG is then computed by dividing the DCG by the ideal DCG (IDCG at k) which is obtained using the above equation but by sorting based on the true labels (rather than ranking based on the model).

IDCG obtained from true labels.

The steps for the rank-loss based adversarial model is shown in Algorithm 2. Note that the rank loss is computed using the current iterate W, it then gets multiplied by the adversarial score derived from the adversarial model A scores, passed through the sigmoid in

	ML20M	Netflix	MSD
Num. users	136,677	463,435	571,355
Num. items	20,108	17,769	41,140
Num. interactions	10M	57M	34M
Validation/Test Size	10k	40k	50k

Table 1: Some characteristics of the datasets used in this paper.

equation (10). We only take the gradient with respect to A based on the rank loss $P(W, A)$ in step 5 of the Algorithm 2. The objective above is not differentiable w.r.t. to W, but is differentiable with respect to A. Non-differentiability with respect to W does not pose any challenges since the gradient ascent steps are only over A. In step 8, we continue to take gradient descent steps with respect to W on $L(W, A)$.

☆

When using a rank loss such as the one shown in the equation above, we refer to the models derived from ARM as R-LARM where R-L stands for rank-loss. Note that the analysis carried in Section 3 also holds for losses similar to the one in equation (10) since the analysis was only over the optimum of the adversary. It directly follows that the adversary would try to reduce the rank loss in regions of high density.

☆

It is straight-forward to extend the above formulation to use a neural network rather than a linear model predicting the adversary scores. For example, we can use one hidden layer $A_0 \in \mathbf{R}^{n \times k}$ with a tanh activation followed by $A_1 \in \mathbf{R}^k$ to get the adversary scores. The regularization is now on both the layers of the neural network as follows. We denote the following formulation as R-LARM-NN.

$$\boxed{\begin{aligned}\max_{A_0, A_1} \sum_{i=1}^{n} &(1 - NDCG_i(k, W))\sigma([tanh(XA_0)A_1]_i)/z \\ &- \frac{B}{2}\|A_0\|^2 - \frac{B}{2}\|A_1\|^2\end{aligned}} \tag{12}$$

The training algorithm for a neural network based adversary models follow the same stesps as in Algorithm 2.

5 EXPERIMENTS

For our experiments, we used datasets that have been previously used in [25] and in [19]. These are 20 million movie lens (ML20M) dataset [11], Netflix Prize dataset (NFLX) and the million song dataset (MSD) [21]. Most of the setup was similar to that used in the previous literature [19, 25]. High level characteristics of these datasets are given in Table 1. All the implementations of the results presented in this paper were done with TensorFlow [1].

The experiment methodology was mostly similar to that used in previous work [25]. We train models on the training data, select any hyper-parameters on the held out validation set and then compute the final results on the held out test data set. Further details on how the individual hyper-parameters were selected can be seen in the sub-sections below.

The above datasets already come with predefined training, validation and test splits. We used the same splits in our experiments as the ones used in previous work [25]. In previous work, three metrics (NDCG@100, Recall@20, Recall@50) were used for evaluations. In this paper, we mostly focus on NDCG. The NDCG@100 metric can detect movements of ranks even within the top 100 unlike recall which is insensitive to rank movements once an item is within the top 100 items. However, the results on all the three metrics were typically highly correlated and we present overall results with all three metrics.

Since all the models were built on top of the closed solution EASE model, the first baseline was the EASE model. We also considered another simple baseline based on Inverse Propensity Weighting (IPW) [20] on top of EASE at the user level. For every user, we counted the number of 1's in their training observations (i.e., the number of movies clicked/viewed), we denote this by θ^i. We then gave a weight $\theta_i^{-\beta}$ for the i^{th} user where β is a hyper-parameter that was tuned. This baseline worked quite effectively compared to the baseline EASE model. With these weights at the user level, we used EASE with closed form solution as described in equation (8).

↳ EASE with IPW

5.1 Implementation Details

We implemented the proposed adversarial learning from Algorithm 1 and Algorithm 2 using stochastic gradient descent/acent using Adam optimizer [17] in Tensorflow [1]. The models W and the adversaries A were initialized with GlorotNormal initializer for all variants. The derivatives were obtained using the auto-gradient functionality of Tensorflow. The objectives in the adversarial models (eqn. (9), (10)) involve a normalization over the entire dataset which would make it hard to implement directly. Instead, we normalized at each mini-batch level. Every mini-batch consisted of 1024 rows of X (corresponding to 1024 users) and all columns. Since MSD is a much bigger dataset, we used a mini-batch size of 8192. Learning rates were tuned between $\{1e-5, 2e-5, 5e-5, 1e-4\}$ for all three datasets. The learning rates were equal for the learner and the adversary. For MSD, the highest learning rate worked well in practice and typically lower learning rates worked better for the other two datasets. NDCG@100 was evaluated after each epoch on the dataset, we performed solver updates until there was a drop from the peak validation NDCG for five epochs or up to a maximum of 100 epochs through the data. Adversarial updates were done after the first epoch (i.e., K>0 in Algorithm 1). The two hyper-parameters C and B for the adversarial models, hidden layer dimensionality for R-LARM(NN) and C and β for the EASE - IPW model were all tuned to achieve best overall performance on the validation set using NDCG@100 metric. Once the best model was selected, we evaluated on the held out test set.

Higher LR worked well for MSD. Lower for other datset

5.2 Results over all users

We first show the overall results on the three datasets in Table 2. For each dataset, every row corresponds to one particular method. Even though hyper-parameters were selected using the validation NDCG, we still report all the metrics that were originally reported in [25]. We were able to exactly match the results from [25] with our own implementation of EASE. As mentioned in Section 4.2, EASE-IPW was also solved using a closed form solution. The remaining two models were solved using versions of Algorithm 1 and Algorithm 2 as described above in Section 5.1. The error bars are given along with the names of the datasets in Table 2. The results show that

Dataset	Model	NDCG@100	Recall@20	Recall@50
ML20M **(0.002)**	EASE	0.420	0.391	0.521
	EASE - IPW	0.424	0.395	0.526
	ARM	0.419	0.389	0.520
	R-LARM	0.425	0.395	0.527
	R-LARM(NN)	0.426	0.396	0.529
NFLX **(0.001)**	EASE	0.393	0.362	0.445
	EASE - IPW	0.398	0.366	0.449
	ARM	0.393	0.361	0.444
	R-LARM	0.397	0.365	0.448
	R-LARM(NN)	0.398	0.365	0.449
MSD **(0.001)**	EASE	0.390	0.334	0.428
	EASE - IPW	0.391	0.334	0.429
	ARM	0.390	0.335	0.428
	R-LARM	0.390	0.335	0.428
	R-LARM(NN)	0.391	0.334	0.429

Table 2: Average results on three public datasets by various methods. The error bars [25] on the three datasets are shown along with the names of the datasets. Both EASE-IPW and R-LARM variants perform significantly better compared to the baseline EASE model. Between EASE-IPW and R-LARM models, the performance is almost comparable. On ML20M dataset, we observe R-LARM(NN) performing significantly better compared to EASE-IPW. The ARM models merely perform comparable to the baseline EASE models with no improvements. A direct comparison with many other methods can be seen in [25] which shows that the performance on all these datasets is near or at the state-of-the-art.

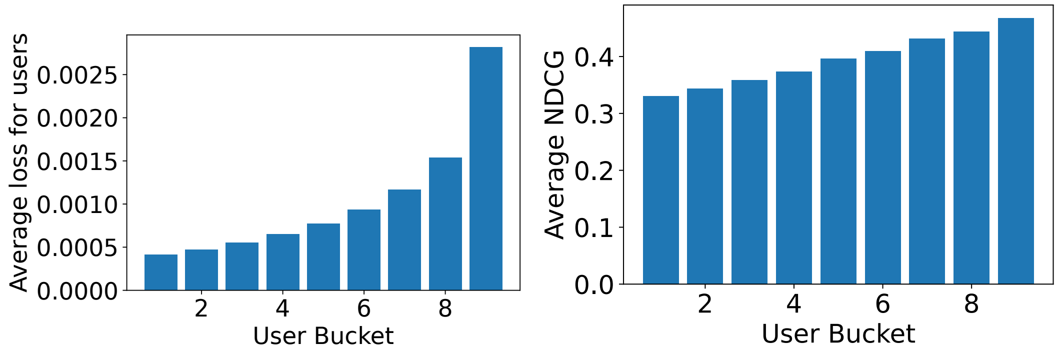

Figure 1: Unusual behavior with ARM. Each bar shows 10% users with increasing number of observations (i.e., 1's) in their X_i. The left plot shows that the average loss at the user level increases with increasing observations, while the right plot shows that NDCG achieved by the EASE model on users also increases with the number of observations per user. These results are with the MSD dataset, but the plots from the other two datasets looked very similar.

ARM barely performs on par with EASE. Both EASE-IPW and R-LARM perform significantly better compared to the EASE baseline. However, between EASE-IPW and R-LARM, there is not too much of a difference compared to their difference from the baseline. R-LARM(NN) performs signficantly better compared to EASE-IPW on ML20M. With these overall results, we dig deeper to understand many aspects of the results.

5.3 Why doesn't the ARM model work effectively?

First, we try to understand why the model ARM in Table 2, which is the most straight-forward extension of the techniques in [18], did not work effectively. The objective for this technique in equation (9) is a least squares loss on a sparse binary matrix. Given that most of the entries in the binary matrix are zero, most of the prediction by the (learner) model are close to zero. Even though the EASE objective results in a good ranking function at user level when ranking all items, for the sake of the adversarial model, the loss behaves strangely when aggregated at the user level. We consider the first term of the objective function in equation (9), i.e.,

$$\sum_{i=1}^{m} \sigma([Xa]_i)/z \sum_{j=1}^{n} (X_{ij} - [XW]_{ij})^2.$$

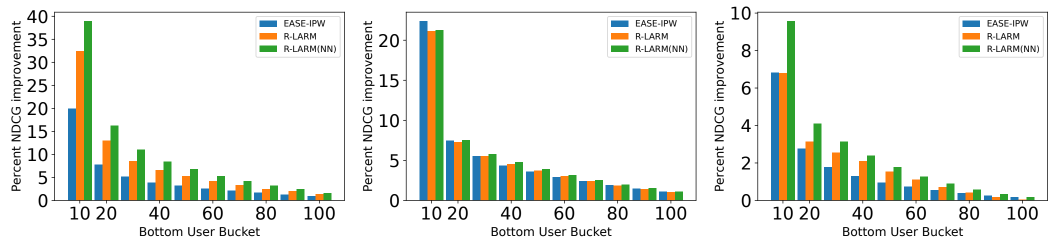

Figure 2: Percentage improvement in NDCG@100 on the three datasets, ML20M, NFLX and MSD from left to right. The x-axis represents buckets of users whose NDCG is within the bottom q percentile with the EASE baseline. The y-axis represents the improvement achieved by EASE-IPW, R-LARM and R-LARM(NN) respectively on the same users with respect to the EASE baseline. For ML20M dataset, the improvements on both R-LARM and R-LARM(NN) were statistically significant by a paired t-test at 5% level on all the bars. On NFLX dataset, results were not significant for R-LARM, results were significantly better on the 4th to 7th bars for R-LARM(NN). For MSD, results were significantly better for R-LARM(NN) on the first eight bars and for R-LARM the differences were significant in all bars except the first two and the last but third.

We observe that the adversarial weight for the i^{th} user is over an aggregated loss over all items with j going from 1 to n. Since the loss is a squared loss and since most of the predictions $[XW]_{ij}$ are small, this loss would be highly correlated with the number of ones for the i^{th} user. Thus, the adversary learns high scores where there are a large number of 1's for a user. However, typically, with more observations for a user, the ranking quality would be higher as well. Thus the adversary does not penalize hard examples when formulated this way; it actually penalizes users with more observations which are typically easier to rank! We illustrate these observations on the MSD dataset but the phenomenon was similar on the other datasets as well. The results are shown in Figure 1. We took the best performing EASE model from Table 2. On the held out test data, we computed NDCG@100 for every user. The users were then grouped into nine buckets. The first bucket being all users with the lowest 10% of 1's in their rows, the next bucket being the next 10% users with the next lowest number of observations etc. The left plot shows that the average user loss being highly correlated with the number of 1's for users. The right plot shows the general behavior of higher NDCG for users with higher observations. These two plots demonstrate that, at a user level, the aggregated loss is highly correlated with the NDCG. Thus directly using the EASE loss at a user level is not appropriate to train an adversary against. This experiment shows that the adversarial model has to be designed carefully with a good understanding how the loss function works.

☆ Why ARM performs poorly

5.4 How do the models differ?

From the results in Table 2, we notice that the R-LARM and EASE-IPW perform quite similarly when we look at the overall performance. R-LARM(NN) also performs comparable to EASE-IPW except on ML20M. A natural question to ask is whether there is any difference between the models learned? To understand this better, we look at users who fall within the bottom q% performance on the test data using the EASE baseline. After selecting the worst-off q% users, we look at the performance of the EASE-IPW model as well as R-LARM and R-LARM(NN) models on the same set of users and compare them both against the baseline EASE model performance. These results are shown in Figure 2 for all the three datasets. For ML20M and MSD datasets, we see that there is significantly large difference between EASE-IPW versus R-LARM and R-LARM(NN) for the users that are not served well (in terms of the model performance) by the baseline model. In the case of NFLX datasets, although there is some improvement on some users, but not as much as on the other two datasets. More details can be found in Figure 2. This result shows that even though we see only slight differences in the overall results, there could be large differences between them on underlying sub-populations. In particular, R-LARM(NN) performs significantly better on disadvantaged populations with respect to the baseline.

→ Key Result

6 CONCLUSIONS

We studied recommender models from an adversarial training view point. We argued that recommender models trained against an adversary tends to reduce loss at high density areas with high loss. The result of such a model training, was empirically shown to be beneficial to under-served users. Our experiments showed that adversarial models can perform near state-of-the-art, yet improving the performance significantly for many users who may be under-served. It is our hope that this type of work would open up broader discussions on how to perform better for under-served users. Although our framework is a general one, our experiments were on top of EASE models. One interesting direction is to consider adversarial models on top of other recommender models. The paper also showed that nuances of the loss function play a significant role in how an adversarial model should be designed. In this paper, we restricted the models to user level adversarial models. Since recommender systems problems have a user and an item angle to them, considering item level or user-item level adversarial models is another important research direction.

ACKNOWLEDGMENTS

The authors thank Harald Steck for helpful discussions about EASE and for the help with the three datasets used in this paper.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Deepak Agarwal, Bee-Chung Chen, Rupesh Gupta, Joshua Hartman, Qi He, Anand Iyer, Sumanth Kolar, Yiming Ma, Pannagadatta Shivaswamy, Ajit Singh, and Liang Zhang. 2014. Activity Ranking in LinkedIn Feed. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. ACM, New York, NY, USA, 1603–1612. https://doi.org/10.1145/2623330.2623362

[3] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. *Bias and Debias in Recommender System: A Survey and Future Directions*. Technical Report. arXiv:2010.03240 https://arxiv.org/abs/2010.03240

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

[5] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2018. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine* 35, 1 (2018), 53–65. https://doi.org/10.1109/MSP.2017.2765202

[6] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2020. *Adversarial Machine Learning in Recommender Systems (AML-RecSys)*. Association for Computing Machinery, New York, NY, USA, 869–872. https://doi.org/10.1145/3336191.3371877

[7] Emily Diana, Wesley Gill, Michael Kearns, Krishnaram Kenthapadi, and Aaron Roth. 2021. *Minimax Group Fairness: Algorithms and Experiments*. Association for Computing Machinery, New York, NY, USA, 66–76. https://doi.org/10.1145/3461702.3462523

[8] John C. Duchi and Hongseok Namkoong. 2021. Learning models with uniform performance via distributionally robust optimization. *The Annals of Statistics* 49, 3 (2021), 1378 – 1406. https://doi.org/10.1214/20-AOS2004

[9] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.* 6, 4, Article 13 (Dec. 2015), 19 pages. https://doi.org/10.1145/2843948

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative Adversarial Networks. *Commun. ACM* 63, 11 (oct 2020), 139–144. https://doi.org/10.1145/3422622

[11] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[12] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018. Fairness Without Demographics in Repeated Loss Minimization. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1929–1938. https://proceedings.mlr.press/v80/hashimoto18a.html

[13] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommendation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 355–364. https://doi.org/10.1145/3209978.3209981

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[15] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-Based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 843–852. https://doi.org/10.1145/3269206.3271761

[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM '08)*. IEEE Computer Society, USA, 263–272. https://doi.org/10.1109/ICDM.2008.22

[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/ARXIV.1412.6980

[18] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed H. Chi. 2020. Fairness without Demographics through Adversarially Reweighted Learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 62, 13 pages.

[19] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. https://doi.org/10.1145/3178876.3186150

[20] R. Lieb M. Höfler, H. Pfister and H. Wittchen. 2005. The use of weights to account for non-response and drop-out. *Social psychiatry and psychiatric epidemiology* 40 (2005).

[21] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. 2012. The Million Song Dataset Challenge. In *Proceedings of the 21st International Conference on World Wide Web* (Lyon, France) *(WWW '12 Companion)*. Association for Computing Machinery, New York, NY, USA, 909–916. https://doi.org/10.1145/2187980.2188222

[22] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 497–506. https://doi.org/10.1109/ICDM.2011.134

[23] J. Rawls. 2001. *Justice as fairness: A restatement.* Harvard University Press.

[24] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 111–112. https://doi.org/10.1145/2740908.2742726

[25] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3251–3257. https://doi.org/10.1145/3308558.3313710

[26] Harald Steck. 2020. Autoencoders That Don't Overfit towards the Identity. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1644, 11 pages.

[27] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. 2020. Adversarial Training Towards Robust Multimedia Recommender System. *IEEE Transactions on Knowledge and Data Engineering* 32, 5 (2020), 855–867. https://doi.org/10.1109/TKDE.2019.2893638

[28] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2021. *Combating Selection Biases in Recommender Systems with a Few Unbiased Ratings*. Association for Computing Machinery, New York, NY, USA, 427–435. https://doi.org/10.1145/3437963.3441799

[29] Hongyi Wen, Xinyang Yi, Tiansheng Yao, Jiaxi Tang, Lichan Hong, and Ed H. Chi. 2022. Distributionally-Robust Recommendations for Improving Worst-Case User Experience. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 3606–3610. https://doi.org/10.1145/3485447.3512255

[30] Chen Wu and Ming Yan. 2017. Session-Aware Information Embedding for E-Commerce Product Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 2379–2382. https://doi.org/10.1145/3132847.3133163

[31] Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. 2021. To be Robust or to be Fair: Towards Fairness in Adversarial Training. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 11492–11501. https://proceedings.mlr.press/v139/xu21b.html

[32] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. DARec: Deep Domain Adaptation for Cross-Domain Recommendation via Transferring Rating Patterns. *CoRR* abs/1905.10760 (2019). arXiv:1905.10760 http://arxiv.org/abs/1905.10760

[33] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 764–773. https://proceedings.mlr.press/v48/zheng16.html