Enhancing Collaborative Filtering with Generative Augmentation

Qinyong Wang
The University of Queensland
qinyong.wang@uq.edu.au

Hongzhi Yin*†
The University of Queensland
h.yin1@uq.edu.au

Hao Wang*
Alibaba AI Labs
cashenry@126.com

Quoc Viet Hung Nguyen
Griffith University
quocviethung1@gmail.com

Zi Huang
The University of Queensland
huang@itee.uq.edu.au

Lizhen Cui
Shandong University
clz@sdu.edu.cn

ABSTRACT

Collaborative filtering (CF) has become one of the most popular and widely used methods in recommender systems, but its performance degrades sharply for users with rare interaction data. Most existing hybrid CF methods try to incorporate side information such as review texts to alleviate the data sparsity problem. However, the process of exploiting and integrating side information is computationally expensive. Existing hybrid recommendation methods treat each user equally and ignore that the pure CF methods have already achieved both effective and efficient recommendation performance for active users with sufficient interaction records and the little improvement brought by side information to these active users is ignorable. Therefore, they are not cost-effective solutions. One cost-effective idea to bypass this dilemma is to generate sufficient "real" interaction data for the inactive users with the help of side information, and then a pure CF method could be performed on this augmented dataset effectively. However, there are three major challenges to implement this idea. Firstly, how to ensure the correctness of the generated interaction data. Secondly, how to combine the data augmentation process and recommendation process into a unified model and train the model end-to-end. Thirdly, how to make the solution generalizable for various side information and recommendation tasks. In light of these challenges, we propose a generic and effective CF model called AugCF that supports a wide variety of recommendation tasks. AugCF is based on Conditional Generative Adversarial Nets that additionally consider the class (like or dislike) as a feature to generate new interaction data, which can be a sufficiently real augmentation to the original dataset. Also, AugCF adopts a novel discriminator loss and Gumbel-Softmax approximation to enable end-to-end training. Finally, extensive experiments are conducted on two large-scale recommendation datasets, and the experimental results show the superiority of our proposed model.

† Contributing equally with the first author.
* Corresponding authors.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330873

KEYWORDS

Adversarial Training; Collaborative Filtering; Data Sparsity

ACM Reference Format:
Qinyong Wang, Hongzhi Yin*†, Hao Wang*, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. 2019. Enhancing Collaborative Filtering with Generative Augmentation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330873

1 INTRODUCTION

While there is an unprecedented number of products and services available on online platforms, it is challenging and time-consuming for users to manually search and find the ones that they are interested in. Recommender systems provide an excellent solution for solving this information overloading problem, i.e., they help the platforms to generate a personalized list containing the items that the target user is most likely to consume.

Among various recommendation techniques, the most widely adopted and well-studied one is collaborative filtering (CF), which assumes that similar users share similar interests (user-based CF) and similar items have similar characteristics (item-based CF). If a CF method only relies on the original user-item interaction data, which is usually represented by a matrix, we call it a pure CF method. Pure CF is quite efficient, scalable and can achieve satisfactory results for active users with sufficient past interactions. However, the user-item interaction matrix is usually very sparse in real applications. Most users can only access a limited number of or even none items, as a result, pure CF can hardly capture those users' preferences and thus suffer from significant performance degradation. This problem becomes especially serious for newly created online services. To alleviate the data sparsity problem in pure CF, hybrid CF is a popular solution, which exploits and integrates additional sources of information about the users or items (also known as the side information). The side information could be obtained from user profiles and item description information such as user demographics and item features. Based on the unique characteristics of side information in different types of datasets, different specific hybrid CF methods were proposed. For example, [39, 43, 45] utilize item's spatial information for POI recommendation, [10] leverages various video metadata such as title, description and raw video streams for Youtube video recommendationand and [25] leverages the social trust paths for better recommendation results. However, these methods have two major flaws. Firstly, the simple and plain

pure CF is already able to achieve good performance for active users, but hybrid CF methods treat each user equally and apply the expensive process of exploiting the side information to all users, incurring extra unnecessary computational resources. Secondly, these methods are carefully designed for specific datasets and tasks, but can hardly generalize to other scenarios without tedious modifications. These limitations and strengths of pure and hybrid CF motivate us to bring up a novel solution which should have the following two features. The first is that it could fully utilize the available user- or item-related auxiliary information to generate high-quality and reliable interaction data only for those inactive users to improve the performance of pure CF methods. Another one is that it could also be trained end-to-end, that is, it uses only one unified process to augment existing user-item interactions and perform pure CF.

To this end, we propose an end-to-end model called **AugCF**. AugCF is based on the recently proposed Conditional Generative Adversarial Nets (CGAN) [31], an important variant of the prominent generative model Generative Adversarial Nets (GAN) [13]. GAN utilizes the minimax game theory to generate plausible fake data samples and has shown excellent performance in a wide range of data generation areas such as images [2], videos [38] and texts [46]. Compared to the vanilla GAN that has no control on modes of the data being generated, CGAN could direct the data generation process by conditioning the model on additional information besides a random prior noise input. Most CGAN-based models naturally consider the class as an additional input to generate class-aware data. Considering the properties of personalized recommendation, our model additionally conditions on both a class (like or dislike) and a user. Specifically, there are two components in AugCF: generator and discriminator. For the generator, it is fed with a sampled user (inactive users preferred), some randomly selected unvisited items of this user, associated side information and a sampled class. Then, a well-trained generator is expected to output the most plausible item for this user under this class, and the generated (*user, item, class*) tuple could be considered as a valid ground-truth augmented data sample to the original dataset. Here the classes and users are additionally considered as extensions to the learned latent space for better generation and discrimination [31]. Also, the ability of the generator to receive different types of side information to generate augmentation makes AugCF a quite generic model to handle many sorts of application scenarios. During training, the discriminator is essentially a classifier that plays different roles in a two-phase procedure. In the first training phase (or data augmentation phase), the discriminator only focuses on discriminating whether an item is fake from the generator or is real from the original dataset. In this phase, the generator and the discriminator are trained in an adversarial manner conditioning on the same user and the same class until an equilibrium state is reached, where the discriminator can no longer distinguish whether an item is fake or real. Then we proceed to the second training phase (or pure CF phase). In this phase, the well-trained generator is fixed and only used as a data provider while the discriminator acts as a pure CF that takes data points from both sources. It is trained to predict an item as liked or disliked by a user, without needing to tell whether it is fake or real. To enable the discriminator with two roles, we follow [47] to introduce a new labeling system that extends from two labels (i.e., like and dislike) to four labels (i.e., real&like, real&dislike, fake&like and fake&dislike). In this way, we can train the discriminator by forcing it to map a training instance into different labels depending on our goal. For example, in the first phase, when the model is conditioned on "this user likes an item", a generated item should be mapped into the "fake&like" label, and a real item should instead be mapped into the "real&like" label. In the second phase, when an item is liked by a user, regardless of its source, should be mapped into both "fake&like" and "real&like".

However, before we can enjoy the benefits of the end-to-end training, we have to overcome the discrete item sampling problem in the generator because it would block the gradient flow, causing backpropagation training invalid. This is one of the major issues when GAN is applied in the discrete data domain [12]. Many existing works attempt to utilize the policy gradient strategy (REINFORCE) [37] to address this non-differentiation problem [41, 46], unfortunately, the variance of its estimated gradients scales linearly with the number of items [34], and this problem becomes especially serious for the recommendation task that needs to deal with millions of items. Recently, Gumbel-Softmax [20] was proposed to overcome these limitations. It introduces a continuous distribution on the simplex that can approximate categorical samples, whose parameter gradients can be easily computed via the reparameterization trick. Gumbel-Softmax has been proved effective in many similar problems such as [36] and [24]. Therefore we also adopt this method to build a bridge between the generator and the discriminator, enabling AugCF to be trained via backpropagation.

Overall, our contributions could be summarized as follows:

- We propose a generic and effective CF model AugCF that supports a wide variety of recommendation tasks. To the best of our knowledge, we are the first to employ Conditional GAN to generate high-quality augmented interaction data to enhance the pure CF methods.
- We propose an end-to-end training procedure for AugCF, which includes a novel discriminator loss and Gumbel-Softmax approximation to overcome the gradient block problem, which has been seldom explored in recommender systems.
- We implement AugCF in two concrete application scenarios and perform extensive experiments. The experimental results show that our proposed AugCF model outperforms the state-of-the-arts.

2 MODEL DESCRIPTION

2.1 Preliminaries

First, we introduce some important notations used throughout the paper. Let $\mathcal{U}$ and $\mathcal{V}$ be the sets of all users and items, and let M and N be their sizes, respectively. We define the interaction between user u and item v in the dataset C as:

$$c = \begin{cases} 1, & u \text{ likes } v; \\ 0, & u \text{ dislikes } v; \\ \emptyset, & \text{unvisited} \end{cases} \tag{1}$$

For some datasets, the feedback is represented as rating, but we could follow the widely adopted method introduced in [17] to transform them into binary representations. For example, a 5-star rating

in the MovieLens dataset [1] can be transformed into 1 if it is 4-5 and 0 otherwise. For u's unvisited items in C (denoted as $\emptyset$), we can employ CF methods to infer their values.

CF methods roughly fall into two categories: pure CF methods and hybrid CF methods. Pure CF methods aim to model users' preferences on items solely utilizing C. The state-of-the-art pure CF methods are based on latent representations, i.e., they project users and items into a shared latent space by using a vector of latent features to represent a user or an item, and their similarity could be estimated by inner product or other metrics [18]. Following [16], we consider pure CF as a prediction task. Given a user $u \in \mathcal{U}$ and an item $v \in \mathcal{V}$, the score for u likes v is estimated by $f(u, v, c; \mathbf{P}, \mathbf{Q})$, where f is a non-linear function and can be implemented by a deep neural nets, $\mathbf{P}$ and $\mathbf{Q}$ are matrices to map users and items into a joint latent space, respectively. $\mathbf{P}$ and $\mathbf{Q}$ are left out to make the notations more compact when no ambiguity arises. Then, the probability of $c = 1$ is measured by the Sigmoid function:

$$P(c = 1) = \frac{\exp(f(u, v, c = 1))}{1 + \exp(f(u, v, c = 1))} \tag{2}$$

However, accurately estimating f usually suffers from the data sparsity problem, i.e., C is usually rather sparse, as a result, pure CF cannot capture all users' preferences, especially for the inactive users who have fewer interaction histories than a predefined threshold. On the other hand, for each user and item in C, there might be some associated side information (a.k.a. auxiliary information or contexts), denoted as $\mathcal{S}$, such as user's gender, purchase time, item image and user's textual review, and we use $\mathcal{S}_{uv} \in \mathcal{S}$ to denote all side information associated with user u and item v. Hybrid CF methods consider both C and $\mathcal{S}$ to improve the expressive power, but they model all users including active and inactive ones in the same way, which would cost extra computational resources. Moreover, they are usually more complex and less efficient than pure CF methods. Therefore, we aim to generate sufficient "real" interaction data to augment C for the inactive users with the help of side information $\mathcal{S}$, and then a pure CF method could be performed on this augmented dataset effectively and efficiently.

2.2 Model Overview

In this section, we briefly introduce our proposed AugCF model. AugCF has two major components: the generator (denoted as G_θ) and the discriminator (denoted as D_ϕ) with parameters θ and ϕ respectively. The generator as a data augmentor can generate high-quality and reliable (u, v, c) tuples while the discriminator plays two roles in the model. First, it distinguishes between a real tuple sampled from C and a fake tuple generated by G_θ. Then, it further acts as a pure CF method to predict whether u likes a given v or not. To train AugCF, we utilize a two-phase procedure. The aim of Phase I is to obtain a good data augmentor, where the generator and the discriminator play a minimax game and eventually reach an equilibrium where the generator generates a tuple that the discriminator can hardly discriminate. Then in Phase II, the parameters of the generator are fixed and it only acts as a data augmentor, while the discriminator is trained to correctly map a user-item pair to a ground-truth label based on data from both the original dataset C and the well-trained G_θ. The structure of AugCF is shown in Figure 1.

Therefore, a crucial step is to devise a novel unified labeling system for the discriminator to simultaneously support the two different training objectives in Phase I and Phase II. Inspired by [47], we introduce four distinct labels (denoted as y) to denote different ground truths of training instances. To avoid confusion, we distinguish between the terms of "class" and "label" for the rest of the paper, i.e., "class" is denoted as c and is a binary indicator (Eq. 1) while "label" is denoted as y and has four values. When an item v is sampled from the real data, we use $y = 1$ or $y = 0$ to represent that user u likes it ($c = 1$) or not ($c = 0$). Similarly, when an item v is from the generator (i.e., fake item), we instead use $y = 3$ or $y = 2$ to represent the same meaning. Next, we briefly introduce how this labeling system supports the two-phase training. In Phase I, since we always train the generator and discriminator under the same class and user (i.e., c is fixed), the label y that discriminator needs to map an input item into is confined to $\{0, 2\}$ when $c = 0$, or $\{1, 3\}$ when $c = 1$, depending on where v is from. In Phase II, we no longer have the constraint for a fixed class, hence, for a given interaction from either source, the discriminator just behaves like a normal pure CF method except that it would map a (u, v) pair into both $y = 0$ and $y = 2$ if u dislikes v, otherwise, it is mapped into both $y = 1$ and $y = 3$. As such, we extend the original pure CF score function f introduced above to enable this multi-label objective, in this way, the score for label i can be measured by $f(u, v, y)$ where $y \in \{0, 1, 2, 3\}$, and Sigmoid in Eq. 2 is replaced by Softmax:

$$P(y = i) = \frac{\exp(f(u, v, y = i))}{\sum_{j=0}^{3} \exp(f(u, v, y = j))} \tag{3}$$

2.3 Phase I: Data Augmentation

In Phase I, the discriminator and the generator compete with each other conditioning on the same class (i.e., $c = 0$ or $c = 1$) and the same user u. Formally, the training objective of Phase I can be simulated as a minimax game [13]:

$$\begin{aligned} \theta^*, \phi^* = \min_\theta \max_\phi (&\mathbb{E}_{(u,v,y)\sim P_C(v|u,c)} \log[D_\phi(v, y|u, c), y \in \{0, 1\}] \\ &+ \mathbb{E}_{(u,v,y)\sim P_{G_\theta}(v|u,c)} \log[D_\phi(v, y|u, c), y \in \{2, 3\}]) \end{aligned} \tag{4}$$

where $P_{G_\theta(v|u,c)}$ denotes the distribution conditioning on u and c generated by G_θ from which a fake item can be sampled, $P_C(v|u, c)$ is the data distribution conditioning on u and c from which a real item is sampled, and D_ϕ will be introduced shortly.

In this minimax game, the discriminator D_ϕ aims to maximize this objective function by learning from labeled data. It guides the generator by mapping the input item into the correct label conditioning on the given class and user. Meanwhile, G_θ acts as a challenger who constantly pushes the discriminator to its limit. Specifically, it generates plausible items conditioning on the given class and user in order to deceive the discriminator, so it aims to minimize this objective function. The generator and the discriminator are trained alternately, and these two players can eventually reach the equilibrium [12]. In our model, the equilibrium is reached when the discriminator is unable to correctly discriminate between the real and fake item. The training details of the two components in Phase I are introduced as follows.

[1] https://grouplens.org/datasets/movielens/

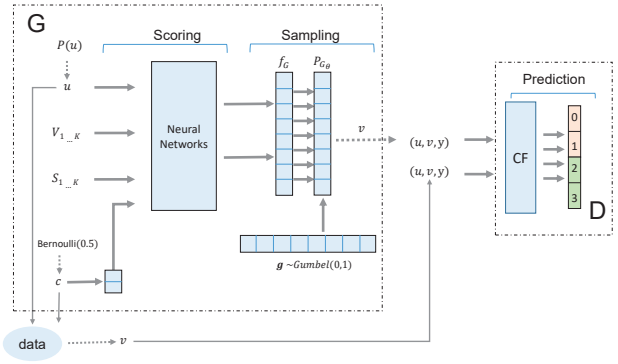


Figure 1: AugCF Overview

2.3.1 Discriminator. Given an item under a user and a class, the discriminator needs to discriminate whether it is real or fake. According to Eq. 4, the objective for training the discriminator in Phase I is:

$$\phi_I^* = \max_{\phi}(L_D^I) \tag{5}$$

where

$$L_D^I = \mathbb{E}_{(u,v,y)\sim P_C(v|u,c)} \log[D_\phi(v,y|u,c), y \in \{0,1\}] + \mathbb{E}_{(u,v,y)\sim P_{G_\theta(v|u,c)}} \log[D_\phi(v,y|u,c), y \in \{2,3\}]$$

where $D_\phi(v,y|u,c)$ estimates whether an item is fake or real conditioning on a specific class c and a user u, which is achieved by constraining the probability P to c and u:

$$D_\phi(v,y|u,c) = P(y|v;u,c) = \frac{\exp(f(u,v,y))}{\sum_{i=0}^{3}\exp(f(u,v,y=i))} \tag{6}$$

2.3.2 Generator. The generator aims to obtain a fake but plausible interaction record, and this process contains the following steps. First, we sample a user u from $\mathcal{U}$ based on this subsampling strategy:

$$P(u) = 1 - \sqrt{\frac{t}{h(u)}} \tag{7}$$

where $h(u)$ is the frequency of user u (i.e., number of u's interactions) and t is a predefined threshold. In this way, the less active users have higher probabilities to be sampled. This is a nice property we want because we prefer to generate more augmentation for the inactive users. This sampling strategy was proposed in [30] and has been proved to work effectively, and we empirically set t to 10^{-5} following [30]. Then, a class $c \in \{0,1\}$ is sampled from a prior Bernoulli distribution parameterised by 0.5. After that, we need to provide items as input to the generator. Though it would be ideal to sample from all unvisited items by u, the number is usually too huge, making it rather inefficient to process them all at a time. We only randomly sample K of them, denoted as $\mathcal{V}_u = \{v_1, \ldots, v_K\}$, and their associated side information is denoted as $\mathcal{S}_u = \{\mathcal{S}_1, \mathcal{S}_1, \ldots, \mathcal{S}_K\}$. Following these steps, u, c, $\mathcal{V}_u$ and $\mathcal{S}_u$ are fed into G_θ, obtaining a point-wise probability distribution $P_{G_\theta(v_i|u,c)}$:

$$P_{G_\theta(v_i|u,c)} = \frac{\exp f_G(u,v_i,\mathcal{S}_i,c)}{\sum_{j=1}^{K}\exp f_G(u,v_j,\mathcal{S}_j,c)} \tag{8}$$

where f_G is the score function for each candidate item that has various implementations based on the application to fully exploit the side information, and we will provide two application scenarios in the next section. After that, an item v is sampled from the probability distribution P_{G_θ}. Furthermore, according to the settings introduced in Section 2.2, we denote this tuple as (u,v,y) for training the discriminator, where y is set as $c+2$ to label this generated item v as fake.

Now that the generator aims to deceive the discriminator that the tuple is real, i.e., to make the discriminator believe that the label $y \in \{0,1\}$, we define the objective function for the generator as:

$$\begin{aligned}\theta^* &= \min_{\theta}((\mathbb{E}_{(u,v,y)\sim P_C(v|u,c)} \log[D_\phi(v,y|u,c), y \in \{0,1\}] + \\ &\quad \mathbb{E}_{(u,v,y)\sim P_{G_\theta(v|u,c)}} \log[D_\phi(v,y|u,c), y \in \{2,3\}]) \\ &= \max_{\theta}(\underbrace{\mathbb{E}_{(u,v,y)\sim P_{G_\theta(v|u,c)}} \log[D_\phi(v,y|u,c), y \in \{0,1\}]}_{\text{denoted as } L_G^I})\end{aligned} \tag{9}$$

2.4 Phase II: Pure Collaborative Filtering

Once Phase I finishes, we perform Phase 2 of training. In this phase, the parameters of the generator are fixed, and it only acts as a reliable data provider. The discriminator is assumed to be unable to tell the source of the provided data. Therefore, we only train the discriminator to predict the class for a given (u, v) pair, which is essentially what a pure CF method does.

Based on the ground-truth labels in Section 2.2, the discriminator in this phase is optimized by:

$$\phi^* = \max_{\phi}(\underbrace{\mathcal{L}_R + \mathcal{L}_G}_{\text{denoted as } L_D^{II}}) \tag{10}$$

where,

$$\mathcal{L}_R = \mathbb{E}_{(u,v,y)\sim P_C} \log[D_\phi(y|u,v) + D_\phi(y'|u,v), y \in \{0,1\}, y' \in \{2,3\}]$$

,

$$\mathcal{L}_G = \mathbb{E}_{(u,v,y)\sim P_{G_{\theta^*}}} \log[D_\phi(y|u,v) + D_\phi(y'|u,v), y \in \{2,3\}, y' \in \{0,1\}]$$

and

$$D_\phi(y|u,v) = P(y|u,v) = \frac{\exp(f(u,v,y))}{\sum_{i=0}^{3}\exp(f(u,v,y=i))}$$

2.5 End-to-End Training

In Phase 1, we notice that G_θ generates an item by sampling from a multinomial distribution, but this operation is not differentiable w.r.t. the distribution parameters [13]. This is one of the major challenges when applying the vanilla GAN and their variants to discrete data because they are usually proposed to generate continuous data such as images and videos. There exist two lines of work to address this challenge. The first line of work adopts the policy gradient strategy (REINFORCE) [37] for optimization [5, 41, 46], which considers the discrete data generation action as a stochastic policy and applies a policy strategy to estimate the generator gradients. However, the policy gradient strategy suffers from large variance because it depends on the sampled trajectories from the current policy to make Monte Carlo estimation. As a result, these methods are usually unstable in practice. The second line of work emerges recently. Many discrete GAN-based models address this challenge

Algorithm 1: AugCF Training Algorithm

INPUT: Training data C and side information $\mathcal{S}$, pre-trained generator G_θ, number of epochs in Phase I and Phase II and batch size J;
OUTPUT: An augmented C and G_{θ^*} and D_{ϕ^*};
for *each epoch in Phase I* **do**
  $\mathcal{B}_{gen} = \emptyset$; $\mathcal{B}_{real} = \emptyset$; //sets for generated and real data
  Sample a batch of users $u_1, \ldots, u_J$ based on Eq. 7;
  Sample a batch of binary classes $c_1, \ldots, c_J \sim Bern(0.5)$;
  for *every* u, c *in batch* **do**
    G_θ generates a distribution $P_{G_\theta(v|(u,c))}$ over K unvisited items based on Eq. 8;
    Sample a K-dimensional $\mathbf{g} \sim Gumbel(0, 1)$;
    Obtain an approximate fake item v based on Eq. 12;
    Construct a fake tuple (u, v, y) where $y = c + 2$;
    Add this (u, v, y) to $\mathcal{B}_{gen}$;
    Randomly sample an item v from C given u and c;
    Construct a real tuple (u, v, y) where $y = c$;
    Add this (u, v, y) to $\mathcal{B}_{real}$;
  end
  update G_θ and D_ϕ with $\mathcal{B}_{gen}$ and $\mathcal{B}_{real}$ based on Eq. 13;
end
for *each epoch in Phase II* **do**
  $\mathcal{B}_{gen} = \emptyset$; $\mathcal{B}_{real} = \emptyset$; //sets for generated and real data
  G_{θ^*} generates a batch of tuples, adding to $\mathcal{B}_{gen}$;
  Add a batch of tuples sampled from C to $\mathcal{B}_{real}$;
  Update D_ϕ with both $\mathcal{B}_{gen}$ and $\mathcal{B}_{real}$ based on Eq. 14;
end

by utilizing Gumbel-Softmax distribution[20, 27], which is a continuous approximation to a multinomial distribution parameterized in terms of the softmax function. In this way, the optimization could be performed by gradient descent directly.

We adopt the second approach for optimization to improve the training stability. Specifically, let $\mathbf{g}$ be a K-dimensional noise vector, where $g_1, \ldots, g_K$ are i.i.d sampled from *Gumbel(0, 1)* [2]. We then obtain the sampled item $\mathbf{v}$ in one-hot representation, i.e., the position of v in this one-hot vector is 1 while the other elements are 0, using the arg max operation using the Gumbel-Max trick [28]:

$$\mathbf{v} = \texttt{one\_hot}\left(\arg\max_i \left[\log P_{G_\theta}(v_i) + g_i\right]\right) \tag{11}$$

Since the arg max operation, again, is not differentiable, the continuous and differentiable softmax function is replaced to approximate it, which is called Gumbel-Softmax [20]. Finally, we obtain an approximate one-hot representation of the sampled item $\mathbf{v}$, i.e.,

$$v_i = \frac{\exp((\log P_{G_\theta}(v_i) + g_i)/\tau)}{\sum_{j=1}^{K} \exp((\log P_{G_\theta}(v_j) + g_j)/\tau)} \qquad \text{for } i = 1, ..., K \tag{12}$$

where τ is a hyper-parameter called temperature, and when it approaches 0, samples from the Gumbel-Softmax distribution become one-hot and the Gumbel-Softmax distribution becomes identical to the multinomial distribution P_{G_θ}. In our implementation, τ is initialized as 0.9, over the course of training, it decays at the rate of 0.3×10^{-4} until it is 0.5, which achieves a good balance between the hard and soft arg max function.

In this way, the whole process can be differentiable, and the model could be optimized with a standard backpropagation algorithm such as SGD.

It is worth to note that we first pretrain the generator independently before it is trained adversarially with the discriminator. Then, for Phase I, we update θ and ϕ by:

$$\theta \leftarrow \theta + \nabla L_G^I \quad ; \quad \phi \leftarrow \phi + \nabla L_D^I \tag{13}$$

For Phase II, we keep θ fixed and only update ϕ with data from both G_θ and C by:

$$\phi \leftarrow \phi + \nabla L_D^{II} \tag{14}$$

The detailed training process of AugCF is shown in Algorithm 1.

3 APPLICATIONS

In this section, we apply our AugCF to two specific scenarios based on the type of the side information: textual contents and sparse features. In other words, we choose two feasible implementations of the score function f_G in Eq. 8 depending on the side information available.

3.1 Content-Based AugCF

Some datasets such as the Amazon dataset [3] and the Yelp dataset [4] contain reviews from users for their consumed items, which could be very helpful to capture item properties and user behaviors. Therefore, it is intuitive to facilitate these textual contexts to tackle sparsity. To sample an item v_i from $\mathcal{V}_u$ using side information $\mathcal{S}_u$ (textual reviews, in this case) conditioning on user u and class c, we implement the generator in our AugCF with a state-of-the-art content-based recommendation method DeepCoNN [48] which models users and items jointly using the textual reviews. Specifically, it learns hidden latent features for users and items based on convolutional neural networks (CNN) [8]. One of the networks models user behavior using the reviews written by user u, and the other network models item properties using the written reviews for every item in $\mathcal{V}_u$. Then the learned latent features for user and item are used for prediction through a layer on the top of both networks, which is motivated by matrix factorization (MF) techniques [22] to let latent factors of users and items interact with each other [48]. Moreover, recall that we additionally consider the class c, so we concatenate its latent representation, which is also learned by a neural network (as shown in Figure 1), to $\hat{\mathbf{z}}$.

Therefore, the score function f_G in AugCF is defined as:

$$f_G(u, v_i, \mathcal{S}_i, c) = \hat{w}_0 + \sum_{i=1}^{|\hat{z}|} \hat{w}_i \hat{z}_i + \sum_{i=1}^{|\hat{z}|} \sum_{j=i+1}^{|\hat{z}|} \left\langle \hat{\mathbf{b}}_i, \hat{\mathbf{b}}_j \right\rangle \hat{z}_i \hat{z}_j \tag{15}$$

where, according to [48], $\hat{\mathbf{z}}$ is a vector that concatenates the learned user, item and class latent representation, $\hat{w}_0$ is the global bias, $\hat{w}_i$ models the strength of the i_{th} variable in $\hat{z}$ and $\left\langle \hat{\mathbf{b}}_i, \hat{\mathbf{b}}_j \right\rangle = \sum_{k=1}^{|\hat{z}|} \hat{\mathbf{b}}_{i,k} \hat{\mathbf{b}}_{j,k}$. $\left\langle \hat{\mathbf{b}}_i, \hat{\mathbf{b}}_j \right\rangle$ models the second order interactions.

We call this application as content-based AugCF.

[2] Gumbel(0,1) can be sampled with $g = -\log(-\log(\mu))$, where $\mu \sim Uniform(0, 1)$

[3] http://jmcauley.ucsd.edu/data/amazon

[4] https://www.yelp.com/dataset/challenge

3.2 Sparse Feature-Based AugCF

The side information in some datasets presents in very sparse formats, that is, almost all of the elements of a training vector are zero, which usually results from dealing with large categorical variable domains. For example, we could transform each example in the music recommendation dataset Last.fm [3] into a sparse feature vector $\mathbf{x} = \{\mathbf{x}_0, \mathbf{x}_1 \ldots\}$ where $\mathbf{x}_i$ stands for the categorical representation of user ID, tags, timestamps, etc. Similarly, we aim to sample an item v_i from $\mathcal{V}_u$ using sparse features $\mathcal{S}_u$ conditioning on user u and class c. However, under such circumstance, it is quite challenging to fully utilize the information to generate augmentation because there is usually not enough data to estimate interactions between variables directly and independently [33]. One of the most effective neural network-based solutions to address this challenge is the Wide & Deep learning framework [7]. It jointly trains wide linear models and deep neural networks to combine the benefits of them for recommender systems, where wide linear models can effectively memorize sparse feature interactions using cross-product feature transformations while deep neural networks can generalize to previously unseen feature interactions through low-dimensional embeddings [7]. For the fixed class c and user u that also needs to be considered as input, we see them as normal categorical features to interact with other original features.

In this case, the score function f_G in AugCF thus is defined as:

$$f_G(u, v_i, \mathcal{S}_i, c) = \mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b \qquad (16)$$

where $\phi(\mathbf{x})$ are the cross product transformations of some important sparse features, and b is the bias term. $\mathbf{w}_{wide}$ is the vector of all wide model weights, and $\mathbf{w}_{deep}$ are the weights applied on the final activations $a^{(l_f)}$.

We call this application as sparse feature-based AugCF.

4 EXPERIMENTS

In this section, we introduce the experiments for the two real-world applications using AugCF, i.e., content-based AugCF and sparse feature-based AugCF. For simplicity, they share the same evaluation metric and procedures. Since each of the two applications has its unique backgrounds, datasets and baseline methods, we report them separately. Finally, we present a case study for a deeper understanding of our proposed model.

4.1 Evaluation Methodology

In this work, we adopt the methodology *Hits@k* (or *Recall@k*) that has been widely used to evaluate the performance of recommender systems such as [9, 19, 42, 44].

First, we use 70% of the whole dataset as the training set and use the remaining as the test set $\mathcal{T}^{test}$. For the test set, we only keep the positive interaction records from the source dataset, e.g., only those ratings larger than 3 stars in the Movielens dataset. Then, for every item v rated positively by user u in $\mathcal{T}^{test}$:

(i) We randomly select R additional items unvisited by user u, where R varies for different datasets. These items can be safely assumed to be uninterested to user u due to the large number of total items. Moreover, the items "visited" by user u in the generated data are also excluded to be sampled.

Table 1: Basic Statistics of Frappe and Movielens

Dataset	#Users	#Items	#Interactions	#Features	Sparsity
Electronics	192,403	63,001	1,689,188	N/A	0.014%
Health	38,609	18,534	346,355	N/A	0.048%
Beauty	22,363	12,101	198,502	N/A	0.073%
Games	24,303	10,672	231,780	N/A	0.089%
Movielens	6,040	3,706	1,000,209	9,812	4.47%
Frappe	957	4,082	288,609	5,382	7.39%

(ii) We predict u's preference scores for these R items and v using the well-trained model. For our AugCF, we only use the discriminator for prediction;

(iii) We form a ranked list by ordering all the R+1 items according to their scores. Let r denote the position of v within the list and thus the best result happens when v precedes all the other items (i.e, r = 1).

(iv) We pick the k items with the highest scores to form a top-k recommendation list. If $r \leq k$, we get a *hit* (i.e., the ground truth v appears in u's recommendation list), otherwise, we get a *miss*.

The *Hits@k* is finally computed as $\frac{\#hit@k}{|\mathcal{T}^{test}|}$, where $\#hit@k$ is the number of hits within test set $\mathcal{T}^{test}$. As the hit probability would rise as k increases or R decreases, we need to fix k and R for all comparison methods to ensure fairness.

4.2 Content-Based AugCF

4.2.1 Datasets. To evaluate this application, we use the well-known and publicly available Amazon product review datasets mentioned above, which cover user interaction (review, rating, helpfulness votes, etc.) on 24 product categories. We pick four of the categories with different characteristics (i.e., sparsity levels and sizes): Electronics, Health and Personal Care ("Health" for short), Beauty and Video Games ("Games" for short). Their basic statistics are listed in Table 1. In this experiment, we set R as 500.

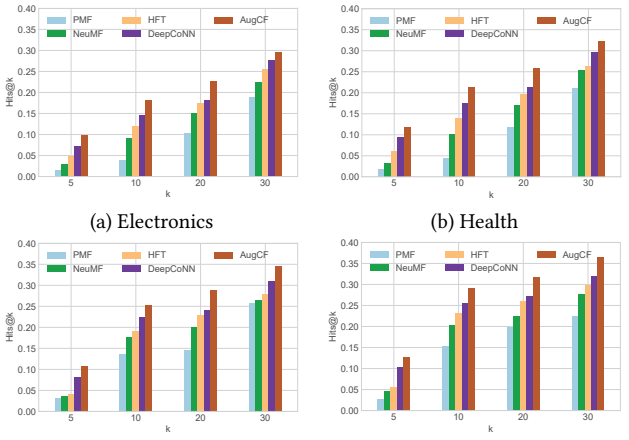
4.2.2 Baselines. We compare our proposed AugCF with four state-of-the-art CF recommendation models, two of which are hybrid CF methods that leverage user reviews and the others are pure CF methods that only consume interaction data.

DeepCoNN [48]: DeepCoNN is a hybrid CF that exploits the information existing in the reviews for recommender systems, and it achieves the state-of-the-art recommendation performance on datasets with reviews. We use the full version of DeepCoNN, and the widely used pretrained Google News word embeddings.

HFT [29]: Hidden Factor as Topic incorporates user reviews into a rating prediction model. Specifically, it combines Matrix Factorization and Latent Dirichlet Allocation [4] to simultaneously train on the ratings and the texts of the reviews. We use 50 latent topics in the experiments.

PMF [32]: Probabilistic Matrix Factorization models the conditional probability of latent factors given the observed ratings and includes Gaussian priors that handle complexity regularization. PMF is one of best known pure CF methods.

NeuMF [16]: 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. NeuMF is also a pure CF method.



(a) Electronics (b) Health

(c) Beauty (d) Games

Figure 2: Evaluation on Four Amazon Datasets

4.2.3 Experimental Results & Analysis. We show the experimental results on Figure 2. From the results, we can see that our method consistently outperforms the competitors on all four datasets, moreover, all differences between our solution and others are statistically significant ($p < 0.01$). Besides, we can reach the following conclusions, which validate many of our claims. First, all methods have better performance if the datasets are denser, i.e., they all obtain the highest hit accuracy on the "Game" dataset while the lowest hit accuracy on the "Electronics" dataset. Then, the hybrid CF models (HFT and DeepCoNN) that furthermore consider the user reviews have better performance than the pure CF methods (PMF and NeuMF), that is why our model is designed to take the side information into consideration. Finally, our model outperforms HFT and DeepCoNN, because we utilize the side information in a novel and effective strategy, that is, the side information is used for generating augmentation to the interaction data, so that we can apply a pure CF method to all users.

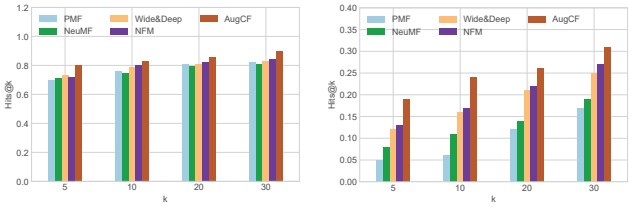
4.3 Sparse Feature-Based AugCF

4.3.1 Datasets. We use the following two datasets that have multiple sparse features for evaluation, and their statistics are also listed in Table 1. In this experiment, we set R as 2000.

Frappe[5]: Frappe is a context-aware app discovery tool. The data contains 8 context factors, which are "daytime", "weekday", "isweekend", "homework", "cost", "weather", "country" and "city". These factors, as well as the user ID and app ID, are converted into feature vectors using one-hot encoding. We adopt the extended version of the dataset used in [14] that adds some negative ratings to the original datasets. In AugCF, we consider "cost" as an item related feature while the rest as user-related features.

Movielens: Movielens provides datasets containing the movie ratings by users. We select Movielens-1M in this experiment, because besides user and movie IDs, it also contains user demographic information: age, gender and occupation, and 18 movie genres. Similarly, we denote every feature (including each genre) as a one-hot feature vector. Similarly, in AugCF, we consider the genres as item

[5]http://baltrunas.info/research-menu/frappe



(a) Frappe (b) Movielens

Figure 3: Evaluation on Frappe and Movielens Datasets

related features and user demographic information as user-related features.

4.3.2 Baselines. For comparison, besides the pure CF methods PMF and NeuMF introduced above, we adopt another two state-of-the-art recommendation methods that can deal with the sparse features in the datasets.

Wide & Deep [7]: As introduced above, Wide & Deep has wide linear models and deep neural networks, combining these two learning techniques enables the recommender to capture both memorization and generalization. According to the literature, this model can improve the accuracy as well as the diversity of recommendation.

NFM [14]: Neural Factorization Machines is one of the state-of-the-art methods based on deep learning for item recommendation, which combines hidden layer of GMF and MLP to learn the user-item interaction function. Also, it replaces the second-order interactions with MLP and proposes regularizing the model with dropout and batch normalization.

4.3.3 Experimental Results & Analysis. The results are shown in Figure 3, from which we can see that, again, the results of our AugCF are better than the baseline methods in both datasets, and are statistically significant ($p < 0.01$). The first and second conclusions from the previous experiment also apply here. Besides, we have the following additional observations. First, these two datasets are denser than the previous Amazon product reviews, thus all methods have higher hit rates on them without surprise. Then, the reasons "Frappe" gets high hits rates are twofold: 1) the "Frappe" dataset is quite dense (see Table 1); 2) the original "Frappe" dataset is an implicit dataset, meaning that it only has positive interactions, the authors of [14] added two times more negative interactions to the original dataset to form the dataset we use, which makes the prediction easier than the true dataset, especially when we also consider the context information. This is another example that requires a more sophisticated method to augment the original dataset. Finally, Wide & Deep and NFM have comparable performance on both dataset, and the reason might be that the higher-order interactions among features are not significant in these datasets.

4.4 Case Study

4.4.1 Usability of Generated Data. In this study, we evaluate whether the generated interaction records are "true" enough to achieve enhancement on pure CF settings. Hence, we perform PMF on the three versions of datasets independently: the generated dataset (GD), the original dataset (OD) and the augmented dataset (AD) that combines both GD and OD. We choose the four severely sparse Amazon review datasets for this study, and only evaluate

Table 2: Performance of PMF on OD, GD and AD

Data	Electronics	Health	Beauty	Games
OD	0.1037	0.1154	0.14953	0.1993
GD	0.0914	0.1042	0.1323	0.1787
AD	**0.1517**	**0.1740**	**0.2176**	**0.2414**

Table 3: Performance of AugCF-Full and AugCF-Sep

Method	Electronics	Health	Beauty	Games
AugCF-Full	**0.2277**	**0.2580**	**0.2877**	**0.3177**
AugCF-Sep	0.1953	0.2172	0.2509	0.3021

Hits@20 for simplicity. The results are shown in Table 2. We can see that training PMF solely on the generated data also achieves comparable results as the original data. Furthermore, the hit rates improve significantly when PMF is trained on the augmented data. Therefore, we are confident in the quality of the generated data, and believe that it can be used as ground-truth training data.

4.4.2 Effectiveness of End-to-End Training. To answer the research question "*does the end-to-end training really help improve the performance?*", we design a variant of the full AugCF model (called AugCF-Full in this study) which has a separated CF component after Phase I is completed, and we call this variant AugCF-Sep. In fact, AugCF-Sep can be regarded as a stand-alone NeuMF model, which is trained on the augmented data. For the evaluation protocol, we follow the same setting as the previous study. The results are shown in Table 3, where we can see that AugCF-Full enjoying the benefits of end-to-end training outperforms AugCF-Sep. In addition, AugCF-Sep still outperforms NeuMF (see Figure 2) because it has been trained on the augmented dataset.

4.4.3 Impact of Sample Numbers. Since a user is uninterested in most items, we study the appropriate value of K, the number of items that are used as input to the generator in Phase I to sample a fake item from, in this study. The optimal K should 1) be sufficiently large to include at least one item this user is interested in; 2) enable Gumbel-Softmax to perform effectively. We set the value of K at the range of [200, 1000] to search for the optimal value for different Amazon review datasets, measured by *Hits@20*. The results are shown in Figure 4, indicating that the optimal values of K vary for different datasets. Clearly, a sparser dataset needs a smaller K and vice versus. For example, the optimal K for the "Games" dataset is around 800 while for the "Electronics" dataset is around 400. Note that when K is set as a very large number, the running time of AugCF increases linearly, so we need to balance the trade off.

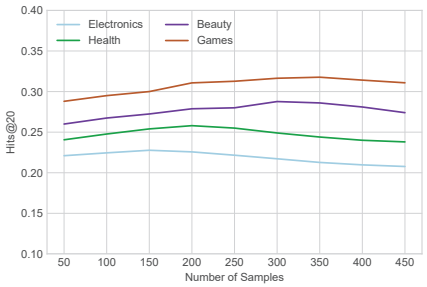


Figure 4: Impact of Number of Samples

5 RELATED WORK

5.1 Data Augmentation

Most data augmentation techniques proposed in the literature can be categorized into transformative or generative methods. Transformative methods synthesize new training samples from the original training samples by applying either additive Gaussian or uniform noise over pre-determined families of transformations, such as [23]. Transformative methods are widely used in the vision domain. For example, methods introduced in [23] are routinely used in classification problems, which create augmentation by image rotation, lighting/color tone modifications, rescaling, cropping, or as simple as adding random noise.

On the other hand, generative methods exploit the generative power of latent variable models to artificially create convincing data samples, and this direction of data augmentation attracts increasing attention after the powerful deep generative models, such as VAE [21] and GAN [13], have been proposed. For example, DA-GAN [1] takes data from a source domain and learns to take any data item and generalize it to generate other within-class data items. However, most generative data augmentation methods focus on continuous data (such as images and videos) but augmenting discrete data using generative models has been rarely explored. Several semi-supervised GAN could also be viewed as augmented unlabeled samples from labeled ones. GraphSGAN [11] is a recent attempt of this line, which was proposed to learn over graphs using GAN with a novel competitive game between generator and classifier, in which generator generates samples in density gaps at equilibrium. Data augmentation has become an essential step to enlarge the training sets to avoid over-fitting in training deep learning models in many domains, but it has been seldom studied in the recommendation area. In fact, to the best of our knowledge, we are the first to leverage generative models to augment the recommendation training dataset, which is typically discrete data.

For recommender systems, another similar line of work is the data imputation technique [26, 35], which aims to infer missing user-item ratings in the rating matrix. However, after imputation, a separate subsequent CF subroutine is needed to produce predictions using the pseudo imputed dataset. Unlike our proposed AugCF, these methods are not end-to-end and require extra manual labor. Moreover, since imputed interactions are not real but inferred, low-quality data could seriously damage the performance of the subsequent CF, but these methods do not have a mechanism to guarantee the reliability of the imputed values.

5.2 GAN for Recommendation

As GAN has demonstrated their abilities and potentials on learning from large (unlabeled) data, it is naturally applied in designing a recommender system most recently. IRGAN [40] makes the first attempt by adopting an adversarial framework to unify the generative and discriminative models in information retrieval. The authors furthermore implement a recommender system under IRGAN. He et al. [15] propose Adversarial Personalized Ranking that utilizes adversarial training to minimize the pairwise BPR objective function. Also, GAN is used to generate difficult adversarial negative

samples to improve pairwise ranking in recommendation [41]. Similar to our work, [6] also employs GAN to address the data sparsity problem in CF following their proposed real-valued, vector-wise training. However, these methods aim to utilize GAN to optimize the loss function while our proposed AugCF uses GAN to directly augment the training data to reduce data sparsity.

6 CONCLUSIONS

Coping with the data sparsity problem when designing a CF method has been a long-standing research topic in the recommender system area. In this paper, we proposed a generic CF model called AugCF to efficiently and effectively alleviate this problem by augmenting the interaction datasets. To obtain reliable user interaction data from rich side information and further form augmented ground-truth training datasets, AugCF is based on Conditional Generative Adversarial Nets to control the generation process. To enjoy the benefit of end-to-end training, we introduced a novel discriminator loss function that can handle different tasks in different training phases. Furthermore, to overcome the gradient block problem while reducing the variance, we adopted the Gumbel-Softmax to optimize the generator. To validate our statements, we applied our proposed AugCF to two scenarios. The experimental results show that our model significantly outperforms some strong and state-of-the-art pure and hybrid CF models.

7 ACKNOWLEDGEMENT

This work was supported by ARC Discovery Projects (Grant No. DP190101985 and No. DP170103954).

REFERENCES

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv* (2017).
[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *ICML*. 214–223.
[3] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *ISMIR*.
[4] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *JMLR* 3, Jan (2003), 993–1022.
[5] Liwei Cai and William Yang Wang. 2017. KBGAN: Adversarial Learning for Knowledge Graph Embeddings. *arXiv* (2017).
[6] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks. In *CIKM*. 137–146.
[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & Deep Learning for Recommender Systems. In *DLRS*. 7–10.
[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *JMLR* 12, Aug (2011), 2493–2537.
[9] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*. 39–46.
[10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and others. 2010. The YouTube video recommendation system. In *RecSys*. 293–296.
[11] Ming Ding, Jie Tang, and Jie Zhang. 2018. Semi-supervised Learning on Graphs with Generative Adversarial Nets. In *CIKM*. 913–922.
[12] Ian Goodfellow. 2016. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv* (2016).
[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.
[14] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.
[15] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommendation. In *SIGIR*. 355–364.
[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[17] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *TOIS* 22, 1 (2004), 5–53.
[18] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *WWW*. 193–201.
[19] Bo Hu and Martin Ester. 2013. Spatial Topic Modeling in Online Social Media for Location Recommendation. In *RecSys*. 25–32.
[20] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. *ICLR* (2017).
[21] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *arXiv* (2013).
[22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 8 (2009), 30–37.
[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In *NIPS*. 1097–1105.
[24] Matt J Kusner and José Miguel Hernández-Lobato. 2016. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution. *arXiv* (2016).
[25] Guanfeng Liu, Yan Wang, Mehmet A Orgun, and Ee-Peng Lim. 2013. Finding K Optimal Social Trust Paths for the Selection of Trustworthy Service Providers in Complex Social Networks. *TSC* 6, 2 (2013), 152–167.
[26] Hao Ma, Irwin King, and Michael R Lyu. 2007. Effective missing data prediction for collaborative filtering. In *SIGIR*. 39–46.
[27] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Ddiscrete Random Variables. *ICLR* (2017).
[28] Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* Sampling. In *NIPS*. 3086–3094.
[29] Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *RecSys*. 165–172.
[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*. 3111–3119.
[31] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *arXiv* (2014).
[32] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic Matrix Factorization. In *NIPS*. 1257–1264.
[33] Steffen Rendle. 2010. Factorization Machines. In *ICDM*. 995–1000.
[34] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*. 1278–1286.
[35] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *KDD*. 713–722.
[36] Zhongchuan Sun, Bin Wu, Yunpeng Wu, and Yangdong Ye. 2019. APL: Adversarial Pairwise Learning for Recommender Systems. *Expert Systems with Applications* 118 (2019), 573–584.
[37] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*. 1057–1063.
[38] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Generating Videos with Scene Dynamics. In *NIPS*. 613–621.
[39] Hao Wang, Yanmei Fu, Qinyong Wang, Hongzhi Yin, Changying Du, and Hui Xiong. 2017. A Location-Sentiment-Aware Recommender System for Both Home-Town and Out-of-Town Users. In *KDD*. 1135–1143.
[40] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*. ACM, 515–524.
[41] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural Memory Streaming Recommender Networks with Adversarial Training. In *KDD*. 2467–2475.
[42] Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Sadiq, and Xiaofang Zhou. 2015. Geo-Sage: A Geographical Sparse Additive Generative Model for Spatial Item Recommendation. In *KDD*. 1255–1264.
[43] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. 2014. LCARS: A Spatial Item Recommender System. *TOIS* 32, 3 (2014), 11.
[44] Hongzhi Yin, Qinyong Wang, Kai Zheng, Zhixu Li, Jiali Yang, and Xiaofang Zhou. 2019. Social Influence-Based Group Representation Learning for Group Recommendation. In *ICDE*.
[45] Hongzhi Yin, Weiqing Wang, Hao Wang, Ling Chen, and Xiaofang Zhou. 2017. Spatial-Aware Hierarchical Collaborative Deep Learning for POI Recommendation. *TKDE* 29, 11 (2017), 2537–2551.
[46] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.
[47] Xiaofeng Zhang, Zhangyang Wang, Dong Liu, and Qing Ling. 2018. DADA: Deep Adversarial Data Augmentation for Extremely Low Data Regime Classification. *arXiv* (2018).
[48] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *WSDM*. 425–434.