Deep Generative Ranking for Personalized Recommendation

Huafeng Liu, Jingxuan Wen, Liping Jing, Jian Yu
Beijing Key Lab of Traffic Data Analysis and Mining
Beijing Jiaotong University
Beijing, China
{huafeng,jingxuan,lpjing,jianyu}@bjtu.edu.cn

ABSTRACT

Recommender systems offer critical services in the age of mass information. Personalized ranking has been attractive both for content providers and customers due to its ability of creating a user-specific ranking on the item set. Although the powerful factor-analysis methods including latent factor models and deep neural network models have achieved promising results, they still suffer from the challenging issues, such as sparsity of recommendation data, uncertainty of optimization, and etc. To enhance the accuracy and generalization of recommender system, in this paper, we propose a deep generative ranking (**DGR**) model under the Wasserstein auto-encoder framework. Specifically, **DGR** simultaneously generates the pointwise implicit feedback data (via a *Beta-Bernoulli* distribution) and creates the pairwise ranking list by sufficient exploiting both interacted and non-interacted items for each user. **DGR** can be efficiently inferred by minimizing its penalized evidence lower bound. Meanwhile, we theoretically analyze the generalization error bounds of **DGR** model to guarantee its performance in extremely sparse feedback data. A series of experiments on four large-scale datasets (*Movielens (20M)*, *Netflix*, *Epinions* and *Yelp* in movie, product and business domains) have been conducted. By comparing with the state-of-the-art methods, the experimental results demonstrate that **DGR** consistently benefit the recommendation system in ranking estimation task, especially for the near-cold-start-users (with less than five interacted items).

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Clustering and classification**.

KEYWORDS

Personalized Recommendation, Deep Generative Model, Bayesian Graphical Model

ACM Reference Format:
Huafeng Liu, Jingxuan Wen, Liping Jing, Jian Yu. 2019. Deep Generative Ranking for Personalized Recommendation. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3298689.3347012

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347012

1 INTRODUCTION

A common task of recommender systems is to improve customer experience through personalized item recommendations based on prior implicit or explicit feedback. Usually, item recommendations focus on predicting a personalized ranking on a set of items that users may like the most. Nevertheless, in real-world scenarios, feedback data is extremely sparse because of the limited user responses and the vast combinations of users and items. In particular, this problem becomes more severe when we handle implicit feedback data, where the unobserved entries (i.e. non-interaction between a user and an item) do not necessarily mean that the item is irrelevant to the user. Thus, how to sufficiently and effectively exploit the statistical characteristics of both observed and unobserved data is a challenging issue for personalized ranking task, especially when the data is extremely sparse.

There is an immense effort in the community of machine learning and statistics to develop probabilistic models of recommendation data, which focus on extracting users' and items' low dimensional latent representations from limited responses by defining various generative processes. From a generative modeling perspective, these models estimate unseen and complex data distributions by characterizing dependency of latent variables, so that the underlying similarities between the users and items can be estimated. Unfortunately, the limited data representation ability hinders these methods to achieve a comparably good result. Consequently, a number of deep neural methods are presented to significantly boost recommendation performance by adding crafted non-linear features into the probabilistic generative models [7, 15].

Most previous work focuses on predicting the rating score of an item given by a user, however, this process deviates from the personalized ranking estimation which is more powerful for top-K recommendation. Meanwhile, the existing deep neural network-based methods suffer from the lack of information in sparse feedback data. Furthermore, the naive integration of deep generative model to collaborative filtering would lead to overfitting, and be stuck in a poor local optima because those deep generative models have been originally applied to the dense information [10, 12]. To date, there is no study about deep generative process in the field of personalized ranking for extremely sparse feedback data. Thus, in this work, we aim to fill the research gap by seamless integrating the implicit feedback data generation and pairwise learning-to-rank process in a deep generative model.

We target on addressing two important issues in personalized ranking task: generalization of the deep generative model on extremely sparse feedback data and uncertainty of the optimizing process. For achieving good recommendation performance on the sparse data, we propose to simultaneously generate the pointwise

feedback data via *Beta-Bernoulli* distribution and create the pairwise ranking list by sufficient exploiting both interacted and non-interacted items for each user. Thus, the proposed model is named as Deep Generative Ranking model (**DGR**). The generalization of **DGR** can be guaranteed in theory. To effectively handle the optimizing process, we adopts *Wasserstein* auto-encoder (WAE) framework to measure the true feedback data distribution and the generative data distribution via *Wasserstein* distance. To the best of our knowledge, this is the first attempt to adapt WAE to personalized ranking problem.

We summarize the major contributions:

- A deep generative ranking model (**DGR**) is proposed for personalized recommendation on sparse feedback data by simultaneously considering pointwise matching and pairwise ranking.
- The *Beta-Bernoulli* distribution is applied to generate the recommendation data from the sampled latent variables, which is more suitable for the sparse implicit feedback data than *Multinomial* distribution used in existing method [15].
- The pairwise ranking response model is designed on the generated data to maximize the margin between positive interacted items and negative or non-interacted items for each user, so that implicit feedback data can be sufficiently used to learn the personalized ranking.
- Wasserstein distance is adopted to measure the true data distribution and the generated data distribution, which is able to sufficiently reduce the reconstruction error.
- By formally defining the generalization of **DGR**, we theoretically demonstrates that its generalization can be guaranteed, which leads to good recommendation performance especially on extreme sparse feedback data.
- The performance of **DGR** was thoroughly investigated on widely-used benchmark datasets in terms of several evaluation metrics, indicating the advantage over the state-of-the-art baselines.

The rest of the paper is organized as follows. The related work is discussed in Section 2. Section 3 gives the proposed deep generative ranking model, the corresponding inferring process and theoretical analysis. In Section 4, a series of experiments on four large-scale datasets (*Movielens (20M)*, *Netflix*, *Epinions* and *Yelp*) are conducted to demonstrate the performance of **DGR** by comparing with the state-of-the-art methods. Finally, a brief conclusion is given in Section 5.

2 RELATED WORK

Research on item recommendation focus on predicting a personalized ranking on a set of items that users may like the most, which is known as collaborative filtering. While early work on CF focused on memory-based approaches, model-based approaches [25], especially factor analysis-based models (e.g., latent factor model) [6, 17], have recently gained attention on point-wise setting due to its superior performance and scalability. This advantage comes from the latent representation that the factor analysis-based CF makes use of the idea of "wisdom of crowds". This latent representation modeling is useful in uncovering the complex patterns of collective users' history and preferences that each user record cannot reveal alone. These work formulated the recommendation task as a regression problem to directly optimize the absolute value of binary ratings, which is collectively known as pointwise recommendation methods. Along another line, research on item recommendation focus on directly optimize the personalized ranking of items for each user. Rendle et al. [19] proposed a pairwise learning method BPR, which optimizes a model based on the relative preference of a user over pairs of items. Ning et al. [18] proposed a novel sparse linear method for top-K recommendations via aggregating from user purchase/rating profiles. Differ from traditional pair-wise ranking algorithm, Shi et al [22] improved top-K recommendation by directly maximizing the mean reciprocal rank (MRR).

Unlike previous linear factor analysis models, whose modeling capacity is limited, recent work focus on non-linear feature learning for item recommendation with the aid of neural networks. Ying et al. [30] proposed a collaborative deep ranking (CDR), a hybrid pairwise approach with implicit feedback, which leverages deep feature representation of item content into Bayesian framework of pairwise ranking model. In order to learn better representation for users and items, Xue et al. [29] proposed a deep matrix factorization (DMF), which uses a two pathway neural network architecture to replace the linear embedding operation used in vanilla matrix factorization. He et al. [7] proposed NeuMF under the neural collaborative filtering (NCF) framework which takes the concatenation of user embedding and item embedding as the input of a multi-layer perceptron (MLP) model to make prediction. To offset the weakness of MLP in capturing low-rank relations, NeuMF unifies MF and MLP in one model. Song et al. [23] combined a designed pairwise ranking classification strategy with neural collaborative filtering framework for ranking estimation.

Different from applying deep neural networks directly to recommendation, researchers are increasingly focusing on combining probabilistic generative process and neural networks for deep generative recommendation models. Salakhutdinov et al. [21] applied restricted Boltzmann machine to CF, and they achieve a superior performance than traditional probabilistic matrix factorization (PMF) [17]. Based on these successes, Li et al. [13] used marginalized denoising auto-encoders to incorporate auxiliary information for both users and items. Wang et al. [28] extended PMF by taking advantage of generative stacked denoising auto-encoder (SDAE) and proposed a unified collaborative deep learning (CDL). In the filed of friend recommendation, a Bayesian personalized ranking deep neural networks (BayDNN) model is proposed via extracting latent structural patterns from social networks [4]. Consider the powerful latent representation ability of VAEs, Li et al. [14] combined PMF and VAE for Top-K recommendation. By replacing original *Gaussian* likelihood with *Multinomial* likelihood, Mult-VAE are proposed to model the generative process of implicit feedback [15]. aWAE [31] extends Wasserstein auto-encoder (WAE) with the aid of mutual information and sparse regularization for collaborative filtering. Though a large amount of deep generative models are proposed, they still suffer from several challenging problems mentioned above, such as sparsity of recommendation data and uncertainty of optimization process.

Thus, in this paper, we focus on simultaneously generating the pointwise feedback data via proper distribution and creating the

pairwise ranking list by sufficient exploiting both interacted and non-interacted items for each user.

3 THE PROPOSED MODEL

In this section, we first present our deep generative ranking (**DGR**) model to simultaneously generate the pointwise implicit feedback data and create the pairwise ranking list of interacted and non-interacted items for each user. The architecture of **DGR** is given in Figure 1. This section second gives the inference learning process of **DGR** model. Next, we formally define the generalization of **DGR** and theoretically demonstrates its generalization error is bounded.

The implicit feedback data is usually represented by a binary matrix $\mathbf{X} \in \mathbb{N}^{n \times m}$ to record the click[1] history among n users and m items. Its element $\mathbf{x}_{ij}$ indicates whether the j-th item is interacted by the i-th user. $\mathbf{x}_i = [\mathbf{x}_{i1}, \cdots, \mathbf{x}_{im}]^\top \in \mathbb{N}^m$ is a binary vector demonstrating the click history of user i on all items. Note that $\mathbf{X}$ could be binarized from user-item rating matrix by assigning 1 to the entries whose rating score is greater than a predefined threshold.

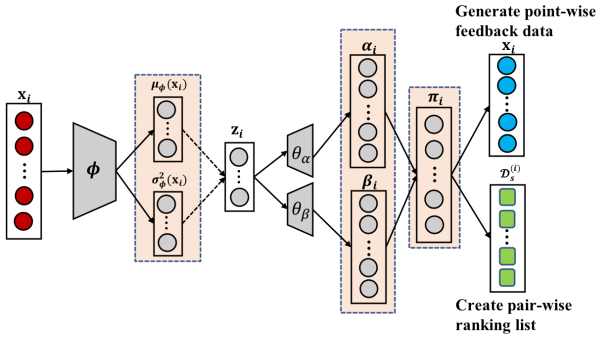


Figure 1: The architecture of the proposed deep generative ranking model.

3.1 Pointwise Feedback Data Generation

This subsection will give the generative process of the pointwise feedback data in **DGR**. To generate the feedback data $\mathbf{x}_i$for user i, a vector of latent variables $\mathbf{z}_i \in \mathbb{R}^k$ (in k-dimensional latent space) is introduced. For convenient computing, $\mathbf{z}_i$ can be sampled from a standard *Gaussian* prior, i.e.,

$$\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_k) \tag{1}$$

To make the generated binary feedback data capturing the preference of the corresponding user to the item, we introduce a vector $\pi_i \in \mathbb{R}^m$ to indicate the extent to which the user i likes the item j, which is also helpful to learn the pairwise ranking list among items (more detail will be given in next subsection). In this case, the *Beta* distribution is adopted to sample π_i because it is a suitable model for the preference score in the range of $[0, 1]$. To generate π_i, **DGR** produces two positive shape parameters $\alpha_i \in \mathbb{R}_+^m$ and $\beta_i \in \mathbb{R}_+^m$ over m items via two non-linear functions $f_{\theta_\alpha}(\cdot)$ and $f_{\theta_\beta}(\cdot)$ from $\mathbf{z}_i$ (in this work, we use a three-layer neural networks activated by *tanh* as the non-linear function and transfer the output via Rectified Linear Units (*ReLu*)). Then π_i can be sampled via

$$\pi_i \sim \prod_{j=1}^{m} \mathcal{B}eta(\alpha_{ij}, \beta_{ij}), \text{ with } \alpha_i = f_{\theta_\alpha}(\mathbf{z}_i) \text{ and } \beta_i = f_{\theta_\beta}(\mathbf{z}_i). \tag{2}$$

Once having the preference score of user over all items, his or her binary implicit data can be modeled by a *Bernoulli* distribution as follows.

$$\mathbf{x}_i \sim \prod_{j=1}^{m} \mathcal{B}(\pi_{ij}) \tag{3}$$

Note that *Bernoulli* distribution is conjugate with *Beta* distribution, which will make the later inference process more efficient. In previous deep generative models, *Multinomial* distribution is always utilized to model the feedback data [3, 15, 31], where the probability vector π_i is constrained by $\sum_{j=1}^{m} \pi_{ij} = 1$). Our model is totally different from the existing work because the proposed *Beta-Bernoulli* distribution samples the preference π_{ij} for each user-item pair and does not assume π_i fall in a $m-1$ simplex. This is reasonable since in real scenarios one user may have multiple interests, the constraint $\sum_{j=1}^{m} \pi_{ij} = 1$ will reduce the corresponding user-item preference. Fortunately, the proposed *Beta-Bernoulli* model can make sure that each user has large probabilities with multiple items if he or she simultaneously and strongly prefers to these items.

Finally, the implicit feedback data can be generated under the *Beta-Bernoulli* distribution with the aid of latent vector $\mathbf{z}_i$ and the non-linear function f_θ as follows.

$$p_\theta(\mathbf{x}_i|\mathbf{z}_i) = \prod_{j=1}^{m} \pi_\theta(\mathbf{z}_i)_j^{\mathbf{x}_{ij}} (1 - \pi_\theta(\mathbf{z}_i)_j)^{1-\mathbf{x}_{ij}} \tag{4}$$

where $\pi_\theta(\mathbf{z}_i)_j$ indicates the generative process from $\mathbf{z}_i$ to π_{ij} and $\theta = \{\theta_\alpha, \theta_\beta\}$ is the parameters of deep neural networks. User i only has one latent representation $\mathbf{z}_i$ over all items and the preference score $\pi_\theta(\mathbf{z}_i)_j$ is independently generated for each item. Thus, the model will put the preference mass on the non-zero entries in $\mathbf{x}_i$, which is consistent with the assumption of *missing not at random (MNAR)* [8, 16] and more suitable to generate the implicit feedback data.

3.2 Pairwise Ranking List Creation

As we mentioned above, the sampling value π_{ij} from *Beta* distribution can be treated as preference score, thus it can be used to create the pairwise ranking list of all items for each user.

Given a feedback dataset, pervious pairwise learning-to-rank method [19] creates tuples (i, j, k) by sampling an observed feedback user-item pair (i, j) and unobserved item k for user i. However, in our real-life applications, there may exist different levels in the users' feedback, such as rating scores with {5,4,3,2,1,0}, likes with {*strongly like, like, unlike, non-interacted*}, etc. Thus, in this case, we split the feedback data into different groups, e.g., positive part (with high priority), negative part (with low priority) and unobserved part (with potential priority).

Specifically, let $\mathcal{P}_p$ denote the positive implicit feedback set, $\mathcal{P}_u$ indicate the unobserved implicit feedback set and $\mathcal{P}_n$ represent the negative implicit feedback set. We define the set order $\mathcal{P}_p \succ \mathcal{P}_u \succ \mathcal{P}_n$, where partial ordered symbol $\succ$ indicates the left set precedes the right set. Thus, we can build the training set $\mathcal{D}_S$ for pairwise

[1]We use the verb 'click' for concreteness to indicate any interaction actions, including "click", "purchase", or "like".

ranking list learning task by

$$\mathcal{D}_S = \left\{(i, j, k)|j \in \mathcal{I}_p^{(i)} \wedge \left((k \in \mathcal{I}_u^{(i)}) \vee (k \in \mathcal{I}_n^{(i)})\right)\right. \tag{5}$$

where i denotes the user together with an item j about which they expressed positive feedback, and an unobserved item or negative item k. $\mathcal{I}_p^{(i)}$, $\mathcal{I}_u^{(i)}$ and $\mathcal{I}_n^{(i)}$ indicate the items set that user i interacted with in positive, unobserved and negative implicit feedback respectively.

Recall that the preference score vector of user i (π_i) has been generated by Eq.(2) in last subsection, it can be used to create the ranking list of all items. If there is a big positive gap between the preference values π_{ij} and π_{ik} (i.e., $\pi_{ij} - \pi_{ik} \gg 0$), it means that user i prefers item j to item k. On the other hand, if there is a big negative gap between π_{ij} and π_{ik} (i.e., $\pi_{ij} - \pi_{ik} \ll 0$), user i prefers item k to item j. In this case, the individual probability of the triple (i, j, k) ($p((i, j, k)|\pi_i)$), i.e., user i really prefers item j to item k, can be defined with the aid of sigmoid function as follows.

$$p((i, j, k)|\pi_i) = \sigma(\pi_{ij} - \pi_{ik}) \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function defined as $\sigma(x) = 1/(1 + e^{-x})$.

Motivated by [19], for user i, we optimize the prediction preference score towards the predefined ground-truth to get the personalized ranking via the following ranking response model

$$\prod_{(i,j,k)\in\mathcal{D}_S^{(i)}} p((i, j, k)|\pi_i) \tag{7}$$

where $\mathcal{D}_s^{(i)}$ indicates the ground-truth set of pairwise ranking triples related to user i.

3.3 Deep Generative Ranking Model (DGR)

By combining *Beta-Bernoulli* distribution generation model and personalized ranking response model, we can obtain the final deep generative ranking model (**DGR**). The first part can be taken as an unsupervised learning on generating the pointwise implicit feedback data. The second part is a supervised learning on pairwise learning-to-rank.

As shown in Figure 1, these two parts share the same generative process from the latent variable $\mathbf{z}_i$ to the predicted probability π_i for each user. Thus, the distribution of the given user feedback data $\mathbf{x}_i$ over all latent variables and parameters for **DGR** can be formulated as

$$p(\mathbf{x}_i|\theta, \mathbf{z}_i) = p_\theta(\mathbf{x}_i|\mathbf{z}_i) \quad \times \prod_{(i,j,k)\in\mathcal{D}_s^{(i)}} p_\theta((i, j, k)|\pi_i) \tag{8}$$

where $\pi_i = \pi_\theta(\mathbf{z}_i)$ is sampled from the latent variable $\mathbf{z}_i$ with the aid of parameter $\theta = \{\theta_\alpha, \theta_\beta\}$. Among it, $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$ indicates the distribution of feedback data in the generation process, which is defined in Eq.(4). $p_\theta((i, j, k)|\pi_i)$ indicates the probability of the individual personalized ranking triple on the predicted score π_i.

3.4 Inference Learning of DGR

One way to look at deep generative models is to postulate that they are trying to minimize certain discrepancy measures between the true data distribution and the generative distribution [5, 10]. In traditional generative model, minimizing the *Kullback-Leibler* (KL) divergence (equivalently maximizing the marginal data log-likelihood) or *Jensen-Shannon* (JS) divergence are commonly-used strategies to estimate the model. However, recommendation data is characterized by few frequently occurring features and a long-tail of rare features, which result in those data exists in a lower dimensional manifold. When directly adopting KL-divergence on high dimensional and sparse recommendation data, a problem we run into is that the generative model may not be optimized fully through pointwise measurements, and usually results in underfitting and fails to utilize the model's full capacity [12]. Thus, we identify a problem with standard deep generative model by utilizing Wasserstein distance for sparse and high-dimensional implicit feedback data [27]. Unlike KL-divergence and JS-divergence, Wasserstein distance has the ability to measure the similarity between the distributions well and preserve the transitivity in latent space since it provides a much weaker topology than KL-divergence and JS-divergence.

To learn the generative model in Figure 1, we are interested in estimating parameter $\theta = \{\theta_\alpha, \theta_\beta\}$ related to deed neural networks. Following [1, 26], we approach generative modeling of **DGR** from the optimal transport point of view. Then the parameter estimation problem can be solved by minimizing certain discrepancy measures between the data distribution $p_{\mathbf{x}_i}$ and model $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$ with the aid of Wasserstein distance [27] under auto-encoder framework. It has been proved that minimizing the Wasserstein distance is equal to optimizing an Evidence Lower BOund (ELBO) [26]:

$$\begin{aligned}\mathcal{L}_G(\mathbf{x}_i; \theta, \phi) = &\inf_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)\in Q} \mathbb{E}_{p_{\mathbf{x}_i}} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)}[c(\mathbf{x}_i, p_\theta(\mathbf{x}_i|\mathbf{z}_i))] \\ &+ \lambda_a \cdot D_{\mathbf{z}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i), p(\mathbf{z}_i)).\end{aligned} \tag{9}$$

The objective is composed of two terms. The first term indicates the reconstruction cost $c(\mathbf{x}_i, p_\theta(\mathbf{x}_i|\mathbf{z}_i))$ measuring the distance between $\mathbf{x}_i$ and $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$. Here we use a squared cost function $c(x, y) = ||x - y||_2^2$ for data points. The second term is a regularizer $D_{\mathbf{z}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i)|, p(\mathbf{z}_i))$ penalizing a discrepancy related to $\mathbf{z}_i$ between prior $p(\mathbf{z}_i)$ and posterior $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$. Here KL divergence is adopted as regularizer, i.e., $D_{\mathbf{z}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i), p(\mathbf{z}_i)) = D_{KL}(q_\phi(\mathbf{z}_i|\mathbf{x}_i), p(\mathbf{z}_i))$. Q is any nonparametric set of probabilistic encoders. We employ the parameterized diagonal *Gaussian* distribution $\mathcal{N}(\mu_\phi(\mathbf{x}_i), diag(\sigma_\phi^2(\mathbf{x}_i)))$ as posterior $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$. λ_a is a hyperparameter controlling the strength of the regularizer.

To establish a personalized ranking loss, we define the negative log-likelihood of the personalized ranking response model (7), i.e.,

$$\begin{aligned}\mathcal{L}_R = -\ln \prod_{(i,j,k)\in\mathcal{D}_S^{(i)}} p_\theta((i, j, k)|\mathbf{z}_i) &= -\ln \prod_{(i,j,k)\in\mathcal{D}_S^{(i)}} p((i, j, k)|\pi_\theta(\mathbf{z}_i)) \\ &= \sum_{(i,j,k)\in\mathcal{D}_S^{(i)}} -\ln \sigma(\pi_\theta(\mathbf{z}_i)_j - \pi_\theta(\mathbf{z}_i)_k)\end{aligned} \tag{10}$$

Our goal is to minimize both the ELBO (9) and the negative log-likelihood (10), which can be implemented by minimizing the final optimization objective

$$\begin{aligned}\mathcal{L}(\mathbf{x}_i; \theta, \phi) = &\inf_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)\in Q} \mathbb{E}_{p_{\mathbf{x}_i}} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)}[c(\mathbf{x}_i, p_\theta(\mathbf{x}_i|\mathbf{z}_i))] \\ &+ \lambda_a \cdot D_{\mathbf{z}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i), p(\mathbf{z}_i)) \\ &+ \lambda_b \cdot \sum_{(i,j,k)\in\mathcal{D}_s^{(i)}} -\ln \sigma(\pi_\theta(\mathbf{z}_i)_j - \pi_\theta(\mathbf{z}_i)_k)\end{aligned} \tag{11}$$

where λ_a is the generative model specific regularization parameter and λ_b controls the effect of personalized ranking.

Note that in feedback data generation process, *Beta* distribution is used as as prior of *Bernoulli* distribution to generate probability π_{ij}. To efficiently implement inference learning, we adopt *reparameterization trick* for sampling from *Beta* distribution [20]. A typical way is to transfer the sampling process for *Beta* distribution via a generalized reparameterization method. Specifically, for probability $\pi_{ij} \sim Beta(\alpha_{ij}, \beta_{ij})$, we could rewrite $\pi_{ij} = \pi_{ij}^{(1)}/(\pi_{ij}^{(1)} + \pi_{ij}^{(2)})$ with $\pi_{ij}^{(1)} \sim Gamma(\alpha_{ij}, 1)$ and $\pi_{ij}^{(2)} \sim Gamma(\beta_{ij}, 1)$. For *Gamma* distribution we choose the transformation

$$\pi_{ij}^{(1)} = \Gamma(\epsilon, \alpha_{ij}) = \exp\left(\epsilon\sqrt{\Psi_1(\alpha_{ij})} + \Psi(\alpha_{ij})\right)$$

where $\Psi(\cdot)$ denotes the digamma function, and $\Psi_1(\cdot)$ is the first-order derivative of Ψ. The random variable ϵ is defined by $\epsilon = \left(\log(\pi_{ij}^{(1)}) - \Psi(\alpha_{ij})\right)/\sqrt{\Psi_1(\alpha_{ij})}$ where $\log(\pi_{ij}^{(1)})$ is the sufficient statistic of $\pi_{ij}^{(1)}$.

Note the parameters need to be optimized are θ and ϕ. We can obtain an unbiased estimate of $\mathcal{L}(\mathbf{x}_i; \theta, \phi)$ by sampling $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{x}_i)$. And by doing *reparameterization trick* for *Gaussian* distribution, we sampling $\epsilon \sim \mathcal{N}(0, \mathbf{I}_k)$ and reparameterize $\mathbf{z}_i = \mu_\phi(\mathbf{x}_i) + \epsilon \odot \sigma_\phi(\mathbf{x}_i)$. The stochasticity of the sampling process is isolated, and the gradient with respect to ϕ can be back-propagated through sampled $\mathbf{x}_i$. Thus we can iteratively update θ and ϕ via stochastic gradient-based optimization techniques.

3.5 Theoretical Analysis

In this subsection, we will theoretically analyze the generalization error of the proposed deep generative ranking model by defining the generalization of **DGR**. According to the generative process of **DGR**, it consists of pointwise implicit data generation and pairwise ranking creation, thus we first analyze them separately and then merge them.

Motivated by [2], we can define the generalization of feedback data generation process by measuring the difference between generative data distribution $\mathcal{X}_g$ and real data distribution $\mathcal{X}_r$, so that the population distance between the true and generative distributions ($\mathcal{X}_r$ and $\mathcal{X}_g$) is close to the empirical distance between the empirical distributions ($\hat{\mathcal{X}}_r$ and $\hat{\mathcal{X}}_g$).

Definition 3.1. For the empirical version of the true distribution ($\hat{\mathcal{X}}_r$) with n training examples, a generated distribution ($\hat{\mathcal{X}}_g$) generalizes under the distance $d(\cdot,\cdot)$ between distributions with generalization error $\delta_1 > 0$ if the following holds with high probability,

$$|E(\mathbf{X}) - E(\hat{\mathbf{X}})| \leq \delta_1$$

where

$$E(\mathbf{X}) = \mathbb{E}_{\mathbf{x}^{(r)} \sim \mathcal{X}_r, \mathbf{x}^{(g)} \sim \mathcal{X}_g} d(\mathbf{x}^{(r)}, \mathbf{x}^{(g)}),$$

$$E(\hat{\mathbf{X}}) = \frac{1}{n}\sum_{i=1}^{n} d(\hat{\mathbf{x}}_i^{(r)}, \hat{\mathbf{x}}_i^{(g)}) | \hat{\mathbf{x}}_i^{(r)} \sim \hat{\mathcal{X}}_r, \hat{\mathbf{x}}_i^{(g)} \sim \hat{\mathcal{X}}_g$$

$E(\mathbf{X})$ indicates the population distance between the true and generated distributions ($\mathcal{X}_r$ and $\mathcal{X}_g$). $E(\hat{\mathbf{X}})$ is the empirical distance between the empirical distributions $\hat{\mathcal{X}}_r$ and $\hat{\mathcal{X}}_g$.

Theorem 3.2. *For any $\mathbf{X} \in \mathbb{N}^{n\times m}$ $(m, n > 2)$,*

$$E(\mathbf{X}) \geq E(\hat{\mathbf{X}}) + \sqrt{\frac{\log \delta_1}{-2n}(\max_i d_i)^2}$$

holds with probability at least $1 - \delta_1$ over uniformly choosing an empirical version of $\mathbf{X}$. *Here, $d_i = d(\hat{\mathbf{x}}_i^{(r)}, \hat{\mathbf{x}}_i^{(g)})$.*

Proof. Since the entries in $\mathbf{X}$ are chosen independently and uniformly, it is reasonable to assume each $d_i = d(\hat{\mathbf{x}}_i^{(r)}, \hat{\mathbf{x}}_i^{(g)})$ is a random variable and satisfies

$$p(\gamma \geq d_i \geq 0) = 1$$

where $\gamma = \max_i d_i$. Hence, based on the Hoeffding Inequality, we have $p(|E(\mathbf{X}) - E(\hat{\mathbf{X}})| \geq \epsilon) \leq \exp(\frac{-2n\epsilon^2}{\gamma^2})$. By setting $\epsilon = \sqrt{\frac{\log \delta_1}{-2n}\gamma^2}$, we have

$$p\left(|E(\mathbf{X}) - E(\hat{\mathbf{X}})| \leq \sqrt{\frac{\log \delta_1}{-2n}\gamma^2}\right) \geq 1 - \delta_1$$

i.e.,

$$p\left(E(\mathbf{X}) \leq E(\hat{\mathbf{X}}) + \sqrt{\frac{\log \delta_1}{-2n}(\max_i d_i)^2}\right) \geq 1 - \delta_1$$

Therefore, the errors of pointwise feedback data generation model are bounded. □

Next, we theoretically analyze the generalization error of the personalized ranking response model. Taking the likelihood of model $\mathcal{L}_R$ (in Eq.(10)) as the evaluation metric, we have the following theorem.

Theorem 3.3. *For any triples set $\mathcal{D}_s$ sampled from expected implicit preference matrix $\mathbf{X} \in \mathbb{N}^{n\times m}$ $(m, n > 2)$ and $\delta_2 > 0$,*

$$L_{\mathcal{D}_s}(\pi) \geq L_{\hat{\mathcal{D}}_s}(\hat{\pi}) + \sqrt{\frac{\log \delta_2}{-2n}\left(\max_i |\hat{\mathcal{D}}_s^{(i)}|\right)^2}$$

holds with probability at least $1 - \delta_2$ over uniformly choosing an empirical version of triples set ($\hat{\mathcal{D}}_s$) from the empirical implicit data $\hat{\mathbf{X}}$. $\hat{\mathcal{D}}_s^{(i)}$ is the triples set related to user i.

Proof. According to Hoeffding Inequality, we have

$$\begin{aligned} p(|L_{\mathcal{D}_s}(\pi) - L_{\hat{\mathcal{D}}_s}(\hat{\pi})| \geq \epsilon) &\overset{(a)}{\leq} \exp\left(\frac{-2n^2\epsilon^2}{\sum_{i=1}^{n}(l_{\hat{\mathcal{D}}_s}(\pi_i))^2}\right) \\ &\overset{(b)}{\leq} \exp\left(\frac{-2n\epsilon^2}{\left(\max_i l_{\hat{\mathcal{D}}_s^{(i)}}(\pi_i)\right)^2}\right) \\ &\overset{(c)}{\leq} \exp\left(\frac{-2n\epsilon^2}{\left(\max_i |\hat{\mathcal{D}}_s^{(i)}|\right)^2}\right) \end{aligned}$$

in which (c) holds due to $p(|\pi_{ij} - \pi_{ik}| \leq 1) = 1$. By setting $\epsilon = \sqrt{\frac{\log \delta_2}{-2n}\left(\max_i |\hat{\mathcal{D}}_s^{(i)}|\right)^2}$, we have

$$p\left(L_{\mathcal{D}_s}(\pi) \leq L_{\hat{\mathcal{D}}_s}(\hat{\pi}) + \sqrt{\frac{\log \delta_2}{-2n}\left(\max_i |\hat{\mathcal{D}}_s^{(i)}|\right)^2}\right) \geq 1 - \delta_2$$

Therefore, the error bound of personalized ranking response model is guaranteed. □

By combining the above two parts, we can calculate the population error and its empirical version of the **DGR** model via

$$E(\mathbf{X}, \pi) = E(\mathbf{X}) + L_{\mathcal{D}_s}(\pi) \text{ and } E(\hat{\mathbf{X}}, \hat{\pi}) = E(\hat{\mathbf{X}}) + L_{\hat{\mathcal{D}}_s}(\hat{\pi})$$

For any $\mathbf{X} \in \mathbb{N}^{n\times m}$ $(m, n > 2)$, triples set $\mathcal{D}_s$, and $\delta > 0$

$$E(\mathbf{X}, \pi) \geq E(\hat{\mathbf{X}}, \hat{\pi}) + \sqrt{\frac{\log \delta}{-2n}\left(\max_i(d_i + |\hat{\mathcal{D}}_s^{(i)}|)\right)^2}$$

holds with probability at least $1 - \delta$ over uniformly choosing an empirical version of $\mathbf{X}$. Thus, the generalization of **DGR** can be guaranteed.

4 EXPERIMENTS

In this section, we evaluate the proposed deep generative model on four datasets by comparing with the state-of-the-art methods.

4.1 Experimental Setting

Datasets: In experiments, four widely used recommendation datasets, *MovieLens 20M (ML 20M)* [2], *Netflix* [3], *Epinions* [4] and *Yelp* [5], are used to validate the item recommendation performance. Among them, *ML 20M* and *Netflix* come from movie domain, *Epinions* belongs to online product domain, and *Yelp* is related to local business domain. The preference scores in *ML 20M* are 10 discrete numbers in the range of [0.5,5] with step 0.5, while the ratings in the other datasets are ordinal values on the scale 1 to 5. Following [15], we binarize explicit data by keeping ratings with four or higher. More detailed information is summarized in Table 1.

Table 1: Summary of experimental datasets

	ML 20M	*Netflix*	*Epinions*	*Yelp*
♯ users (n)	138,493	480,189	49,290	1,182,626
♯ items (m)	26,744	17,770	139,738	156,638
♯ ratings	20,000,263	100,000,000	664,824	4,731,265
RDensity	0.54%	1.17%	0.010%	0.0026%
$\overline{m}_i$	144	208	14	4
$\overline{n}_j$	748	5,627	5	30

$\overline{m}_i$: the average number of items rated by each user
$\overline{n}_j$: the average number of users interested in each item

Metrics for ranking estimation: Recall ($Recall@K(i) = |Re(i) \cap T(i)|/|T(i)|$) is used to test whether the item is in the top-K list, where $Re(i)$ denotes the set of recommended items to user i and $T(i)$ denotes the set of favorite items of user i. Meanwhile, the normalized discounted cumulative gain (NDCG) is adopted to measure the item ranking accuracy, which can be computed by: $NDCG@K(i) = \frac{DCG@K(i)}{IDCG@K(i)}$ (where $DCG@K(i) = \sum_{r=1}^{K}(2^{\mathbb{I}(w(r)\in\mathcal{I}_i)} - 1)/\log_2(r+1)$, and $IDCG$ is the DCG value with perfect ranking). $\mathbb{I}(\cdot)$ is the indicator function and $w(r)$ is the item at rank r. $\mathcal{I}_i$ is the set of items that user i clicked on. For both metrics, larger values indicate better performance. We reported the averaged results and their statistical significances via 5-fold cross-validation technique.

Baselines: We choose three kinds of recommendation methods as baselines, including traditional ranking methods, deep neural network recommendation models, and deep generative recommendation models. The details are summarized as follows:

- Traditional ranking methods: SLIM [18] is a sparse linear method for top-K recommender systems.
- Deep neural network recommendation models: NCF [7] is a neural network-based collaborative filtering method. CDAE [28] is the first model which applies denoising auto-encoder to top-K recommendation problem. NCR [23] is general collaborative ranking framework based on neural networks.
- Deep generative recommendation models: Mult-VAE [15] applies variational auto-encoder to collaborative filtering for implicit feedback. aWAE [31] extends Wasserstein auto-encoders (WAE) for collaborative filtering.

We do not include BPR [19] as baselines since it has been proved to be inferior to deep neural network recommendation methods and deep generative recommendation methods [15, 23].

Parameter setting: The parameters of all baselines are either adopted from their original papers or determined by experiments. For **DGR**, the architecture of the neural networks for encoder and decoder is symmetrically, and they are three-layer perceptrons with architecture $[m \rightarrow 600 \rightarrow k \rightarrow 600 \rightarrow m]$ where m is the number of items. Based on cross validation, we select $k = 150$ for all datasets. Moreover, we find that deeper network does not benefit recommendation. *tanh* is used as the activation function between neural layers. The dropout technique [24] is adopted at the input layer with probability 0.5 and we do not apply the weight decay for any parts. The model is trained using Adam [9] with batch size of 128 users for 200 epoch on all datasets.

4.2 Results and Discussion

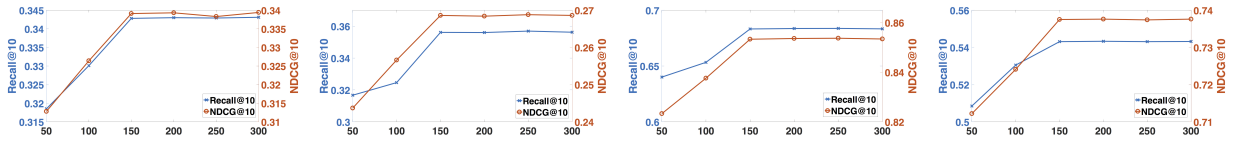
In this section, we investigate **DGR** from two views. Firstly, a series of experiments are conducted to test the effect of parameters. Secondly, **DGR** is compared with baselines from two facets including *All Users* and *Near-cold-start Users* in terms of ranking estimation.

4.2.1 Effect of Parameters. The first experiment is designed to investigate the effect of latent space size k for recommendation performance. Figure 2 shows the results on four datasets in terms of Recall@10 and NDCG@10 under varying k from 50 to 300. As expected, for each dataset, **DGR** performs better as k increases, reaches the best value at around $k = 150$ for all datasets, and it does not improve the performance when the dimension of the latent space is greater than 150. Large k indicates that users will be represented in higher dimensional space, which may not be benefit for recommendation performance since lower space has enough ability to capture the latent properties. Moreover, large k aggravates the computational complexity because more variables are introduced. For instance, the number of weights and biases related to the neural networks from $\mathbf{z}_i$ to α_i and β_i obviously increases with the increasing of k. Results on the other metrics show the same trend, and thus we omit them due to the page limitation.

Figure 3 shows the effect of regularizer parameter λ_a on four datasets in terms of Recall@10 and NDCG@10. The results demonstrate that **DGR** performs better as λ_a increases, reaches the best value at around $\lambda_a = 0.2$ for *ML 20M* and *Epinions* and $\lambda_a = 0.3$ for *Netflix* and *Yelp*, and then decreases in performance as λ_a grows larger. We believe this is because a smaller λ_a has the ability to make use of the relation between prior and posterior for $\mathbf{z}_i$, while a larger λ_a may put too much attention on capacity limitation, thus decrease the efficacy of the recommendation.

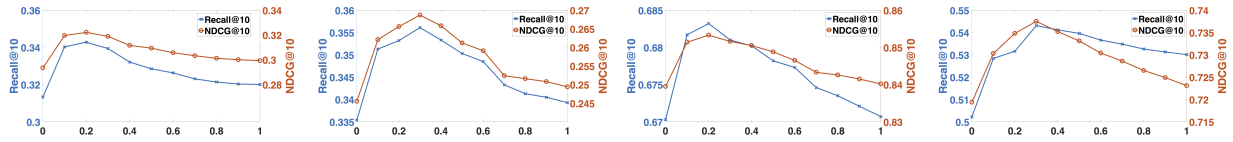
Additionally, the coefficient λ_b in **DGR** controls the contribution of pairwise ranking creation process, which is tuned from 0 to 1 with step 0.1. Figure 4 shows the effect of λ_b on four datasets in terms of Recall@10 and NDCG@10. It can be seen that the recommendation performance becomes better and better as λ_b increases which suggests that the proposed personalized ranking loss is useful to predict the ranking list. The results reach the best when $\lambda_b = 0.2$ for *Yelp*, $\lambda_b = 0.3$ for *Epinions* and $\lambda_b = 0.5$ for *ML 20M* and *Netflix*. After that, the performance decreases, which indicates that large λ_b will make the personalized ranking dominate the learning process. If ignoring the generation of implicit feedback data, in this case,

[2] https://grouplens.org/datasets/movielens/
[3] https://www.netflixprize.com
[4] http://www.trustlet.org/downloaded_epinions.html
[5] https://www.yelp.com/dataset/challenge



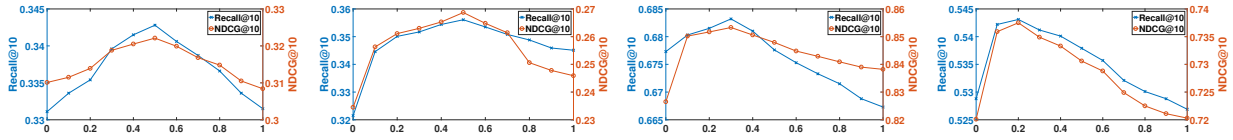
(a) *ML 20M* (b) *Netflix* (c) *Epinions* (d) *Yelp*

Figure 2: Effect of k (the latent space size) in terms of Recall@10 and NDCG@10 on DGR for four datasets.



(a) *ML 20M* (b) *Netflix* (c) *Epinions* (d) *Yelp*

Figure 3: Effect of λ_a (regularization coefficient) in terms of Recall@10 and NDCG@10 on DGR for four datasets.



(a) *ML 20M* (b) *Netflix* (c) *Epinions* (d) *Yelp*

Figure 4: Effect of λ_b (ranking loss coefficient) in terms of Recall@10 and NDCG@10 on DGR for four datasets.

the model may learn inadequate latent representation. Meanwhile, the best λ_b is small for two sparse datasets (*Epinons* an *Yelp*), while large for two dense datasets (*ML 20M* and *Netflix*), which indicates sparse datasets are less confident to construct ranking list.

When $\lambda_b = 0$, the objective function degenerates into Eq.(9) denoted as **DGR-R**, i.e. deep generative model withtout pairwise ranking creation. Table 2 lists the recommendation accuracy obtained by **DGR** and **DGR-R** on four datasets in terms of Recall@10 and NDCG@10, respectively. As expected, **DGR** is superior to **DGR-R**. This result further demonstrates that it is necessary to simultaneously consider pointwise feedback data generation and pairwise ranking creation.

Table 2: Comparisons between DGR and DGR-R.

Metrics	Methods	*ML 20M*	*Netflix*	*Epinions*	*Yelp*
Recall@10	**DGR**	0.3428	0.3561	0.6832	0.5431
	DGR-R	0.3334	0.3421	0.6801	0.5403
NDCG@10	**DGR**	0.3391	0.2687	0.8533	0.7375
	DGR-R	0.3289	0.2549	0.8503	0.7331

Above experiments demonstrate the effect of parameters. Actually, we select optimal λ_a and λ_b via ancestral sampling [15]. Specifically, we start training with $\lambda_a = 0$ and $\lambda_b = 0$, and gradually increase λ_a and λ_b to 1. We linearly anneal the KL term or ranking loss term slowly over a large number of gradient updates to θ and ϕ and record the optimal value when its performance reaches the peak.

4.2.2 Recommendation Performance. In order to deeply investigate the performance of recommendation, the second experiment is conducted to compare the proposed **DGR** with six baselines from two views (*All Users* and *Near-cold-start Users*). *All Users* indicates that all users are used as the testing set. *Near-cold-start Users* view means that the users with less than five interacted items are involved in the testing set.

Table 3, 4, 5 and 6 show the recommendation performance (Recall and NDCG at different numbers of recommended items) for *All Users* on four datasets. The best and second results are marked in bold and underline. Obviously, most of the deep methods (**DGR**, aWAE, Multi-VAE, NCR, NCF, CDAE) significantly outperform traditional ranking approaches (SLIM), which indicates non-linear features are beneficial for improving recommendation quality. As we can see, in most cases, deep generative models (**DGR**, aWAE, Multi-VAE) are superior to other deep methods (CDAE, NCF, NCR), which indicates that proper generative process for poinwise data matching is helpful to learn more useful latent representation. **DGR** performs better than the state-of-the-art deep methods (CDAE, NCF, NCR, aWAE, Multi-VAE), which confirms that considering both generation of implicit feedback and personalized ranking is helpful to predict more precise recommended items.

As we known, sparser feedback data, more challenging the personalized recommendation task. In these four datasets, the average number of *Near-cold-start Users* (with less than five ratings) are 1543, 2201, 34310 and 7439651 in *ML 20M*, *Netflix*, *Epinions* and *Yelp* respectively, and they are about 1.85%, 0.46%, 69.6%, 78.4% of all users in the corresponding datasets. It can be seen that *Epinions* and *Yelp* are extremely sparse, thus we further evaluate the recommendation performance on *Near-cold-start Users* for these two datasets. The results obtained by the proposed model and all baselines on *Epinions* and *Yelp* are listed in Table 7. As expected, **DGR** has the ability to handle sparse problem and is superior to all baselines. The reason is that constructing ranking based triples has the ability to explore more information about ranking list, which confirms that pairwise ranking list creation can be seemly combined with pointwise feedback data generation to effectively improve the recommendation performance.

Table 3: Comparing different recommendation methods on testing *All Users* for *ML 20M* dataset.

Metrics	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
SLIM	0.3122	0.3722	0.5033	0.3033	0.3133	0.3631
CDAE	0.3221	0.3916	0.5231	0.3224	0.3264	0.3654
NCF	0.3332	0.3925	0.5251	0.3258	0.3325	0.3733
NCR	0.3338	0.3936	0.5287	0.3279	0.3336	0.3772
Mult-VAE	0.3320	0.3951	0.5377	0.3185	0.3354	0.3830
aWAE	0.3330	0.3910	0.5320	0.3380	0.3380	0.3690
DGR	**0.3428**	**0.4113**	**0.5444**	**0.3391**	**0.3425**	**0.3862**

Table 4: Comparing different recommendation methods on testing *All Users* for *Netflix* dataset.

Metrics	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
SLIM	0.3003	0.3231	0.4033	0.2126	0.3076	0.3218
CDAE	0.3177	0.3432	0.4285	0.2335	0.3231	0.3401
NCF	0.3223	0.3464	0.4321	0.2456	0.3312	0.3587
NCR	0.3261	0.3501	0.4378	0.2485	0.3305	0.3601
Mult-VAE	0.3208	0.3514	0.4446	0.2449	0.3306	0.3621
aWAE	0.3410	0.3540	0.4441	0.2560	0.3310	0.3651
DGR	**0.3561**	**0.3549**	**0.4472**	**0.2687**	**0.3354**	**0.3689**

Table 5: Comparing different recommendation methods on testing *All Users* for *Epinions* dataset.

Metrics	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
SLIM	0.6413	0.6612	0.7584	0.8257	0.8536	0.8989
CDAE	0.6451	0.6846	0.7732	0.8322	0.8633	0.9021
NCF	0.6785	0.6933	0.7911	0.8465	0.8764	0.9133
NCR	0.6791	0.6943	0.7921	0.8451	0.8746	0.9115
Mult-VAE	0.6812	0.6967	0.7931	0.8489	0.8801	0.9185
aWAE	0.6801	0.6983	0.7954	0.8501	0.8811	0.9201
DGR	**0.6832**	**0.7015**	**0.7983**	**0.8533**	**0.8842**	**0.9246**

Table 6: Comparing different recommendation methods on testing *All Users* for *Yelp* dataset.

Metrics	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
SLIM	0.5027	0.5349	0.5639	0.7144	0.7233	0.7257
CDAE	0.5133	0.5433	0.5771	0.7246	0.7351	0.7385
NCF	0.5278	0.5523	0.5836	0.7314	0.7433	0.7586
NCR	0.5391	0.5525	0.5836	0.7318	0.7426	0.7589
Mult-VAE	0.5322	0.5531	0.5824	0.7328	0.7346	0.7651
aWAE	0.5353	0.5533	0.5864	0.7316	0.7326	0.7634
DGR	**0.5431**	**0.5587**	**0.5960**	**0.7375**	**0.7523**	**0.7754**

Although the numerical improvements are small, small improvements can lead to significant differences of recommendations in practice [11]. To demonstrate the efficiency of the proposed method intuitively, we conduct paired t-test (confidence 0.95) between **DGR** and three baselines with five-fold cross-validation results. The p-values in all cases are less than 10^{-5}, which indicates that the improvements are statistically significant at the 5% level. Therefore, based on these observations, we can say **DGR** consistently outperforms the state-of-the-art methods and significantly improves the recommendation performance.

5 CONCLUSIONS

This paper proposes a deep generative ranking (**DGR**) model for item recommendation that jointly models the generation of implicit feedback data and the creation of pairwise ranking list. The model can be taken as a joint deep generative method that adopts *Beta-Bernoulli* distribution to characterize the feedback data and ranking response model to learn the pairwise ranking list. The generalization of our proposed model is theoretically guaranteed The experiments have shown that **DGR** has ability to output the high-quality recommended item list. In all cases, **DGR** significantly outperforms the start-of-the-art methods, especially for near-cold-start users with extremely sparse feedback data.

Table 7: Comparing different recommendation methods on testing *Near-cold-start Users* for *Epinions* and *Yelp* dataset.

Datasets	*Epinions*		*Yelp*	
Metrics	Recall@10	NDCG@10	Recall@10	NDCG@10
SLIM	0.4233	0.7285	0.2833	0.6887
CDAE	0.4375	0.7344	0.3012	0.6922
NCF	0.4544	0.7435	0.3185	0.7033
NCR	0.4564	0.7441	0.3105	0.7056
Mult-VAE	0.4587	0.7461	0.3244	0.7123
aWAE	0.4597	0.7477	0.3252	0.7161
DGR	**0.4616**	**0.7493**	**0.3287**	**0.7189**

ACKNOWLEDGEMENT

Liping Jing is the corresponding author. This work was supported in part by the National Natural Science Foundation of China under Grant 61822601, 61773050, and 61632004; the Beijing Natural Science Foundation under Grant Z180006; the Beijing Municipal Science & Technology Commission under Grant Z181100008918012.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of ICML*. 214–223.

[2] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. 2017. Generalization and equilibrium in generative adversarial nets (gans). In *Proceeding of ICML*. JMLR. org, 224–232.

[3] Kenan Cui, Xu Chen, Jiangchao Yao, and Ya Zhang. 2018. Variational collaborative learning for user probabilistic representation. *arXiv preprint arXiv:1809.08400* (2018).

[4] Daizong Ding, Mi Zhang, Shao-Yuan Li, Jie Tang, Xiaotie Chen, and Zhi-Hua Zhou. 2017. Baydnn: Friend recommendation with bayesian personalized ranking deep neural network. In *Proceedings of cikm*. ACM, 1479–1488.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*. 2672–2680.

[6] Prem Gopalan, Jake M Hofman, and David M Blei. 2013. Scalable recommendation with poisson factorization. *arXiv preprint arXiv:1311.1704* (2013).

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of WWW*. 173–182.

[8] JosÂťe Miguel HernÂťandez-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Probabilistic matrix factorization with non-random missing data. In *Proceedings of ICML*. 1512–1520.

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: a method for stochastic optimization. In *Proceedings of ICLR*.

[10] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[12] Rahul G Krishnan, Dawen Liang, and Matthew Hoffman. 2017. On the challenges of learning with inference networks on sparse, high-dimensional data. *Proceedings of AISTATS* (2017).

[13] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of CIKM*. ACM, 811–820.

[14] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of SIGKDD*. ACM, 305–314.

[15] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. *Proceedings of WWW* (2018).

[16] Benjamin M. Marlin and Richard S. Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of RecSys*. 5–12.

[17] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Proceedings of NIPS*. 1257–1264.

[18] Xia Ning and George Karypis. 2012. SLIM: sparse linear methods for top-N recommender systems. In *Proceedings of ICDM*.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI*. AUAI Press, 452–461.

[20] Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. 2016. The generalized reparameterization gradient. In *Proceeding of NIPS*. 460–468.

[21] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of ICML*. ACM, 791–798.

[22] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of RecSys*. ACM, 139–146.

[23] Bo Song, Xin Yang, Yi Cao, and Congfu Xu. 2018. Neural collaborative ranking. In *Proceedings of CIKM*. ACM, 1353–1362.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[25] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009).

[26] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2018. Wasserstein auto-encoders. In *Proceedings of ICLR*.

[27] Cédric Villani. 2008. *Optimal transport: old and new*. Vol. 338. Springer Science & Business Media.

[28] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of WSDM*. ACM, 153–162.

[29] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of IJCAI*. 3203–3209.

[30] Haochao Ying, Liang Chen, Yuwen Xiong, and Jian Wu. 2016. Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback. In *Proceedings of PAKDD*. Springer, 555–567.

[31] Jingbin Zhong and Xiaofeng Zhang. 2018. Wasserstein autoencoders for collaborative filtering. *arXiv preprint arXiv:1809.05662* (2018).