# RsyGAN: Generative Adversarial Network for Recommender Systems

Ruiping Yin*†, Kan Li*, Jie Lu†, Guangquan Zhang†
*School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
†Centre for Artificial Intelligence, University of Technology Sydney, Sydney, Australia
Emails: yrp@bit.edu.cn, likan@bit.edu.cn, Jie.Lu@uts.edu.au, Guangquan.Zhang@uts.edu.au

***Abstract*—Many recommender systems rely on the information of user-item interactions to generate recommendations. In real applications, the interaction matrix is usually very sparse, as a result, the model cannot be optimised stably with different initial parameters and the recommendation performance is unsatisfactory. Many works attempted to solve this problem, however, the parameters in their models may not be trained effectively due to the sparse nature of the dataset which results in a lower quality local optimum. In this paper, we propose a generative network for making user recommendations and a discriminative network to guide the training process. An adversarial training strategy is also applied to train the model. Under the guidance of a discriminative network, the generative network converges to an optimal solution and achieves better recommendation performance on a sparse dataset. We also show that the proposed method significantly improves the precision of the recommendation performance on several datasets.**

***Index Terms*—component, formatting, style, styling, insert**

## I. Introduction

Recommender systems have become increasingly important in recent years due to the problem of information overload in e-commerce [1]. The use of recommender systems allows individual searches to be more effective by filtering information. Many companies are also interested in using recommender systems to target their customers and recommend products. Recommender systems model the preferences of users through their click history, purchase records or list of favourites. The recommendation task is to predict missing user-item preferences given the observations of these historic records [2].

Existing methods for recommender systems can be divided into three classes: content-based methods, collaborative filtering (CF) methods, and hybrid methods. Many recommender systems use collaborative filtering methods to make recommendations [3]. The most successful CF methods try to learn latent factors according to user-item interactions such as user-item rating or user purchase history [4].

A severe problem with CF methods is that it is difficult to train the model with sparse datasets. The collaborative filtering approaches, especially matrix factorization methods, rely on factorizing the user-item matrix into two latent factor matrices to represent users and items. However, the factorization could be very non-robust when the user-item matrix is very sparse. It always causes a lower quality local optimum in the experiments.

Many works attempted to address this problem. Several authors have merged models to obtain more robust results. [5] [5] combined latent factor models and neighbourhood models to build a more accurate combined model. [6] employed an autoencoder model to learn latent user preferences. These methods try to learn the latent factors through user-item interactions. But these methods cannot converge to an optimal solution because of the severe sparsity of the dataset, which results in an inability to provide ideal recommendation results.

There are also some methods use auxiliary information. Item content information and an item-tag matrix are combined in collaborative topic regression to address the sparsity problem in [7]. A hierarchical Bayesian model has been proposed to address the auxiliary information sparse problem [8]. However, auxiliary information is unavailable in some scenarios.

In this paper, we develop a generative network and a discriminative network inspired by generative adversarial networks [9] to train a property model for recommender systems. The generative network is able to generate the missing preferences for users and the discriminative network is established to evaluate the generative network and guide training process. The model is trained using an adversarial training strategy.

In the experiments, our model demonstrates significant improvements in performance on common datasets such as movieLens [10] and Taobao. Our main contributions in this paper are as follows:

- We propose a novel recommendation model in which the adversarial training strategy is used for the first time to improve the recommendation quality. We treat recommendation generating as a generative process and utilize a discriminator to help it escape from a lower quality local optimum.
- We develop an efficient adversarial optimization algorithm with two loss functions to ensure that this model can be trained efficiently until it converged.
- We conduct experiments on three real-world datasets to evaluate the effectiveness of our method. Experimental results reveal that our method outperforms six state-of-the-art methods in terms of precision, recall and mean average precision metrics.

The rest of this paper is organized as follows. In Section 2, we review some of the relevant methods. Section 3 presents our approach in detail. The experiments and results analysis

are demonstrated in Section 4. Lastly, conclusions and future work are discussed in Section 5.

## II. Related Work

This section summarizes the existing collaborative filtering approaches and the works on generative adversarial networks.

### A. Collaborative Filtering Methods

Collaborative filtering (CF) methods help people to make choices based on the preferences of other people who share similar interests [1], [2]. Many previous works have concentrated on explicit feedback, e.g. in terms of ratings. Nevertheless, in real-world scenarios, implicit feedback is more common than explicit feedback [11], [12]. Implicit feedback, such as clicks, shares, purchases, can be collected automatically [13]. In this paper, we focus on making recommendations according to implicit feedback.

Collaborative filtering (CF) methods can be divided into two categories [14]: memory-based CF and model-based CF. Memory-based CF is the first-generation CF technique which makes recommendation by calculating the similarity values between users or items [15], [16]. However, this type of CF is unlikely to recommend unpopular items to users. It also takes a long time to make a recommendation when the dimension of the user-item matrix is high. The next generation CF methods are model-based CF [17], [18]. The matrix factorization method has been proven to be efficient and effective in many recommender systems [19], [20]. A non-uniform item sampler has been proposed to address the problem in which the convergence of stochastic gradient descent learning algorithms slows down if the item popularity has a tailed distribution [11]. Because a good recommender particularly emphasizes accuracy near the top of the ranked list, a new pairwise ranking loss has been proposed to reduce computational complexity [21].

In recent years, several efforts have been made to apply deep learning models in recommender systems [22]. Restricted Boltzmann machines (RBM) have been used to make recommendations to reduce the training time [3], [23]. The collaborative deep learning method takes two different sources of information into account by using a hierarchical Bayesian model [8]. The autoencoder framework has been proven to be compact and efficient [6], [24]. A similar method, the collaborative denoising auto-encoder method, was proposed by [25]. [26] propose a matrix factorization model with deep neural network architecture using both explicit rating and non-preference implicit feedback.

However, the sparse nature of dataset leads to a problem: how can we train the models enough with too little data.

### B. Generative Adversarial Networks

Recently, generative adversarial networks (GANs) and GANs-based models have been successful in tasks like image generation, text generation, feature extraction and so on [9], [27]. Based on the adversarial training strategy, the generative network is able to map input variables into another feature space through minimax optimization with the discriminative network. [28] propose a unifying generative and discriminative information retrieval model with minimax game strategy. However, the two networks are completely separate in their model, which results in a failure to effectively capture latent features.

Inspired by the training strategy of GANs, we proposed RsyGAN model and an adversarial training method to solve the insufficient training problem resulted in a sparse dataset and make more reliable recommendations.

## III. Generative Adversarial Network for Recommender Systems

In this section, we first give the problem formulation of the recommendation task. We then introduce our proposed generative adversarial model (RsyGAN) and give details of the loss functions, followed by the model optimization algorithm.

### A. Problem Formulation

In this paper, we focus on making recommendation according to implicit feedback. Suppose there are $M$ users $U = \{u_1, \ldots, u_M\}$, $N$ items $V = \{v_1, \ldots, v_N\}$. Let $R \in R^{M \times N}$ denote the implicit feedback matrix, where $R_{ij}$ equals 1 when interactions exist between user $i$ and item $j$, and 0 otherwise. The formulation $R_{i*}$ can be used to represent a user feature in which some elements are missing. Given a history of user actions, the recommendation task tries to predict a list of items which the user might like. A recommender system is commonly formulated as the problem of estimating the missing values in user feature vector $R_{i*}$.

### B. Proposed Model

Inspired by GAN, we combine a generative network and a discriminative network to train a property model for the recommendation task. Figure 1 illustrates our proposed model.

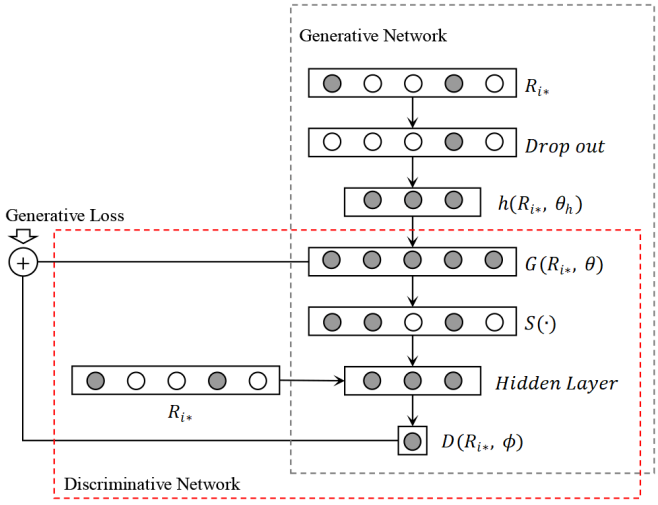


Fig. 1. The architecture of the RsyGAN model

Two networks have been contained in the boxes delineated by a dashed line. The portrait box is a generative network and the landscape box is a discriminative network. The input of the generative network is the user feature vector $R_{i*}$ and the input of the discriminative network is the combination of real user feature vector $R_{i*}$ and the generated user feature by the generative network.

The recommendation task attempts to predict the missing elements in the feature vector. The generative network accepts the feature vector with missing values and returns a similar vector with all missing positions are filled. We can use the output vector to predict the user preferences more accurately.

We apply an autoencoder neural network structure as the generative network. Our network has a number of differences from the classical autoencoder. In this neural network, the feature vector with high dimension will be mapped into a hidden layer which is a lower feature space. The process of dimensionality reduction can be regarded as the extraction of features for user embedding. The hidden layer is computed as follows:

$$h(R_{i*}, \theta_h) = \sigma(W_g \times g(R_{i*}) + b_g) \tag{1}$$

where $\sigma(\cdot)$ is the activation function and $\theta_h = \{W_g, b_g\}$. $g(\cdot)$ is the dropout function. The dropout function is required to avoid over-fitting, because the dataset is too sparse.

We then use an output layer to recover the original user feature vector from the hidden layer. The missing values in the feature vector are filled in the output vector. The output value $\widetilde{R}_{i*}$ can be described as follows:

$$G(R_{i*}, \theta) = \sigma(W'_g \times h(R_{i*}, \theta_h) + b'_g) \tag{2}$$

where $\theta = \{W'_g, b'_g, \theta_h\}$. The reverse mapping may optionally be constrained by tied weights where $W = W'$ in the autoencoder, but different weights are used in our method.

As mentioned in Section 1, we always get a lower quality local optimum due to the sparsity of the dataset. We have therefore designed a discriminative network as an quality indicator of our recommendation model, the generative network. It can be used to help the parameters be trained on the property direction. Because the evaluation function, discriminative network, can be updated according to the convergent recommendation model to help the training algorithm escaping from local optimum.

The discriminative network contains three layers: the input layer, the hidden layer and the output layer. The discriminative network can be described as follows:

$$D(R_{i*}, \phi) = \sigma(W'_d \times \sigma(W_d \times R_{i*} + b_d) + b'_d) \tag{3}$$

where $\phi = \{W_d, b_d, W'_d, b'_d\}$ and $\sigma(\cdot)$ is also the activation function.

We have attempted two activation functions which are defined by the formulae (4) and (5). The impaction of the activation functions is discussed in the experiments.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

$$ReLU(x) = max(0, x) \tag{5}$$

### C. Loss Function

Another key problem is to design a proper objective function according to the input data and output values. In GANs, the generative network samples synthetic data from a hidden feature space represented by a multilayer perceptron. However, it cannot be used directly in the recommendation task, because this task is a prediction problem, therefore we employ the incomplete user history as the input of the generative network. The generative network can only be used to predict missing values, whereas the discriminative network will try to distinguish between real users and users generated by the generative network. Two different loss functions are utilized to conduct the two step optimization.

In the discriminative network training process, we have:

$$J^D = \max_{\phi} \sum_{i}^{M} \log D(R_{i*}, \phi) + \log(1 - D(G^*(R_{i*}, \theta), \phi)) \tag{6}$$

where $G^*(R_{i*}, \theta) = S(G(R_{i*}, \theta))$, $S(\cdot)$ is a sample function with Bernoulli distribution $x \sim B(1, \widetilde{R}_{ij})$. The output of the generative network is a vector containing continuous values in which $\widetilde{R}_{ij} \in (0, 1)$; however, the ground truth is the binary value $R_{ij} \in \{0, 1\}$.

We append $J^D$ into the loss function of the generative network to influence the training of the model. The loss function of the generative network is as follows:

$$J^G = \min_{\theta} \sum_{i}^{M} (\|W_{i*} \circ (R_{i*} - G(R_{i*}, \theta))\|_F^2 + \lambda_D(\log D(R_{i*}, \phi) + \log(1 - D(G^*(R_{i*}, \theta), \phi)))) + \frac{\lambda_\theta}{2} \cdot \|\theta\|_F^2 \tag{7}$$

where $W \in \{0, 1\}^{M \times N}$ is a non-negative weight matrix. Because there are positive examples in missing values, the weight matrix means the confidence of the examples. In our experiments, we set $W_{ij} = 1$ if $R_{ij} = 1$ and a low confidence level $W_{ij} = 0.1$ otherwise.

### D. Optimization Algorithm

Our model contains two parts: the discriminative network and the generative network. The parameters in both networks are initialized randomly before training commences. During the adversarial training stage, the generative network and the discriminative network are trained alternately with Eqs. (6) and (7).

The model for RsyGAN is built using Tensorflow and trained with synchronous stochastic gradient descent updates. We have also open-sourced our implementation on GitHub.

We describe the detailed optimization process in Algorithm 1.

**Algorithm 1** Optimization Algorithm of the Proposed Model

Input: user-item matrix $R$
Output: approximated user-item matrix $\widetilde{R}$
Initialize $G(R_{i*}, \theta)$ and $D(R_{i*}, \phi)$ with random weights $\theta$, $\phi$.
**repeat**
  **for** d-step **do**
    Sample a batch $R_t$ from training set $R$
    Calculate the filled matrix $\widetilde{R}_t$ using $G(R_{i*}, \theta)$
    Sample from $\widetilde{R}_t$ which is subjected to Bernoulli distribution
    Update parameters $\phi$ by using Eq. (6)
  **end for**
  **for** g-step **do**
    Sample a batch $R_t$ from training set $R$
    Update parameters $\theta$ using Eq. (7)
  **end for**
**until** converges

## IV. Experiments

### A. Datasets

Experiments are conducted on three datasets namely MovieLens 1M, MovieLens 10M and Taobao. The basic statistics are listed in Table I. We select 60% of records as the training set. Some records contain explicit feedback such as ratings. As we want to solve an implicit feedback task, we remove the ratings from these datasets.

TABLE I
Statistics of the two datasets

| | users | items | feedback | sparsity |
|---|---|---|---|---|
| ML-1M | 6,040 | 3706 | 939,809 | 0.9580 |
| ML-10M | 69,878 | 10,677 | 104,000,054 | 0.9865 |
| Taobao | 8,349 | 5,701 | 321,976 | 0.9932 |

**MovieLens** is a widely used dataset in many researches, and many versions have been released on the GroupLens website. We choose MovieLens 1M (ML-1M) and MovieLens 10M (ML-10M) to evaluate our method.

**Taobao** is a dataset for competitively matching clothing on the Tianchi platform. It contains basic item data and data on the historical behaviour of users. We use only the historical behaviour data to make recommendations. We remove users with less than 10 items ($|R_{i*}| < 10$) and items with less than 20 users ($|R_{*j}| < 20$) from this dataset.

### B. Evaluation for Recommendation

For top-k recommendation, we evaluate the performance of each approach using metrics precision (Prec@k), recall (Recall@k) and mean average precision (MAP@k).

Given a top-k recommendation result $C_k$, we can compute precision and recall as follows:

$$Precision@k = \frac{\sum_{i=1}^{|U|} |C_{k,i} \cap T_i|}{|U| \times k} \tag{8}$$

$$recall@k = \frac{\sum_{i=1}^{|U|} |C_{k,i} \cap T_i|}{\sum_{i=1}^{|U|} |T_i|} \tag{9}$$

where $C_{k,i}$ is the top-k recommendation list of user i and $T_i$ is the items that user i has adopted in the test set.

Average precision (AP) is a ranked precision metric which is used to score information retrieval. AP@k is the average precision of all positions, which is defined as follows:

$$AP@k = \frac{\sum_{n=1}^{k} Precision@n \times rel(n)}{\min\{k, |T_i|\}} \tag{10}$$

where $rel(n)$ is an indicator function equalling 1 if the item at rank k is contained in the test set, otherwise 0. MAP is the mean of the AP scores for all users.

It is difficult to optimize these metrics directly because they are discontinuous. The loss function in our method is used in learning to approximate these metrics. In our experiments, we mainly show the result of top-k when $k = \{5, 10, 20, 50\}$.

### C. Performance Comparison

In this subsection, we compare the proposed RsyGAN with the methods below. As our proposed model aims to make user recommendations by considering only the relationship between users and items, we mainly compare RsyGAN with user-item models.

- ItemPop: Always recommends the top-k most popular items to users.
- ItemKNN: The classical memory-based collaborative filtering method. Pearson correlation is used in our experiment and the top 50 most similar users are selected as the nearest neighborhood.
- BPR-MF: This is also a content-free algorithm based on matrix factorization which is designed for top-k recommendation tasks [18]. It optimizes pair-wise preferences between observed and unobserved items.
- CDAE: Collaborative denoising auto-encoders [25] learn latent representations of corrupted user-item preferences which can reconstruct the full input. This model is similar to our generative network.
- NCF: Neural network-based Collaborative Filtering (NCF) is a general framework for replacing the inner product with a neural architecture that can learn an arbitrary function from data.
- RsyGAN: Our method proposed in this paper.

We cannot compare our method with RBM because the result of RBM is a binary list. It cannot be evaluated by the metrics in our experiments.

We carefully choose the hyper-parameters for each baseline method. The overall performance of the compared approaches is shown in Tables II.

We can see from the experimental results that RsyGAN achieves significant improvements across all the evaluation metrics and all the datasets. Note that the generative network is similar to CDAE, but we obtain better performance than it does. Our explanation is that there are too many local

TABLE II
RECOMMENDATION PERFORMANCES IN TERMS OF PRECISION AND RECALL

| | | Precision | | | | Recall | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Prec@5 | Prec@10 | Prec@20 | Prec@50 | Recall@5 | Recall@10 | Recall@20 | Recall@50 |
| ml-1m | *POPRANK* | 0.2085 | 0.1911 | 0.1868 | 0.1506 | 0.0742 | 0.1211 | 0.1736 | 0.2530 |
| | *ItemKNN* | 0.2466 | 0.2351 | 0.2263 | 0.2021 | 0.0833 | 0.1367 | 0.1978 | 0.2632 |
| | *BPR-MF* | 0.4932 | 0.4617 | 0.4026 | 0.3224 | 0.0853 | 0.1495 | 0.2149 | 0.3058 |
| | *CDAE* | 0.5699 | 0.5183 | 0.4876 | 0.4592 | 0.0900 | 0.1556 | 0.2472 | 0.3826 |
| | *NCF* | 0.5920 | 0.5222 | 0.4895 | 0.4611 | 0.0920 | 0.1623 | 0.2676 | 0.3974 |
| | *RsyGAN* | 0.6632 | 0.6105 | 0.5087 | 0.3918 | 0.1091 | 0.1775 | 0.2702 | 0.4072 |
| ml-10m | *POPRANK* | 0.1934 | 0.1873 | 0.1628 | 0.1347 | 0.0307 | 0.0743 | 0.1008 | 0.1941 |
| | *ItemKNN* | 0.2404 | 0.2269 | 0.2124 | 0.1817 | 0.0384 | 0.0942 | 0.1392 | 0.2057 |
| | *BPR-MF* | 0.4153 | 0.3892 | 0.3260 | 0.2287 | 0.0612 | 0.1266 | 0.1800 | 0.2928 |
| | *CDAE* | 0.4674 | 0.4118 | 0.3643 | 0.2816 | 0.0752 | 0.1853 | 0.2200 | 0.3733 |
| | *NCF* | 0.4962 | 0.4759 | 0.3993 | 0.3082 | 0.0782 | 0.1832 | 0.2342 | 0.3542 |
| | *RsyGAN* | 0.5333 | 0.4854 | 0.4097 | 0.3007 | 0.0885 | 0.2094 | 0.2480 | 0.4023 |
| Taobao | *POPRANK* | 0.0040 | 0.0036 | 0.0030 | 0.0066 | 0.0024 | 0.0053 | 0.0069 | 0.0283 |
| | *ItemKNN* | 0.0517 | 0.0501 | 0.0466 | 0.0194 | 0.0267 | 0.0376 | 0.0582 | 0.0416 |
| | *BPR-MF* | 0.1047 | 0.0920 | 0.0897 | 0.0544 | 0.0408 | 0.0706 | 0.1370 | 0.2079 |
| | *CDAE* | 0.0889 | 0.0766 | 0.0608 | 0.0367 | 0.0362 | 0.0585 | 0.1196 | 0.1565 |
| | *NCF* | 0.1108 | 0.1023 | 0.0856 | 0.0586 | 0.0492 | 0.0774 | 0.1402 | 0.2152 |
| | *RsyGAN* | 0.1392 | 0.1314 | 0.0937 | 0.0624 | 0.0544 | 0.1038 | 0.1428 | 0.2343 |

minimums in the solution space, and it is very easy to converge to a lower quality local optimum in the CDAE method. The discriminative network can be regarded as a strong constraint for the generative network when the entire solution space is searched.

We also observe that the results on MovieLens are much better than those on Taobao, because Taobao dataset is sparser than the MovieLens.

### *D. Components in RsyGAN*

In this Section, we study the influence of several main components, including the types of activation functions, the number of hidden units, and the hyper-parameter $\lambda_D$.

TABLE III
PERFORMANCE COMPARISON OF THE ACTIVATIONS FUNCTION ON MOVIELENS 1M

| | MAP@5 | MAP@10 | MAP@20 | MAP@50 |
|---|---|---|---|---|
| Sigmoid | 0.4834 | 0.3973 | 0.3420 | 0.3176 |
| ReLU | 0.4561 | 0.3800 | 0.3315 | 0.2939 |

TABLE IV
PERFORMANCE COMPARISON OF THE ACTIVATIONS FUNCTION ON MOVIELENS 10M

| | MAP@5 | MAP@10 | MAP@20 | MAP@50 |
|---|---|---|---|---|
| Sigmoid | 0.4617 | 0.3804 | 0.3292 | 0.2919 |
| ReLU | 0.4401 | 0.3721 | 0.3122 | 0.2881 |

As mentioned in Section 3, we have two different types of activation function. We study their influence separately on two datasets. We show the results for the sigmoid function and the ReLU function on the hidden layer in Tables III, IV and V.

TABLE V
PERFORMANCE COMPARISON OF THE ACTIVATION FUNCTIONS ON TAOBAO

| | MAP@5 | MAP@10 | MAP@20 | MAP@50 |
|---|---|---|---|---|
| Sigmoid | 0.1018 | 0.0869 | 0.0715 | 0.0573 |
| ReLU | 0.0982 | 0.0827 | 0.0706 | 0.0416 |

We can see from the three tables that the sigmoid function performs better than the ReLU method in each case in the experiment, but that ReLU improves more on the larger dataset than on the smaller dataset. One possible cause is that the non-linear part in our model performs better in a small dataset. However, the model with ReLU function can be trained more efficiently, so if we have a very large dataset, we should perhaps choose the ReLU function, and if we need greater precision in a small dataset, we should choose the sigmoid function. To overcome the weakness of the linear activation function, we can also choose multiple layers for model construction.

The number of hidden units is possibly another sensitive parameter in addition to the activation function. In Figure 2, we evaluate the performance of our method as the number of hidden units varies. We observe that the best performance is obtained when the number of hidden units is around 80 on the Taobao dataset and 350 on the MovieLens 10M dataset. This illustrates that the number of hidden units should increase with the increase in the size of the dataset.

Lastly, we study the effect of $\lambda_D$ in our proposed method. We can see in Eqs. (6) and (7) that the value of the loss function of the discriminator network is very small compared to the loss function of the generative function. The output of the discriminative network is just one value range from 0 to 1, and the output of the generative networks have N numbers

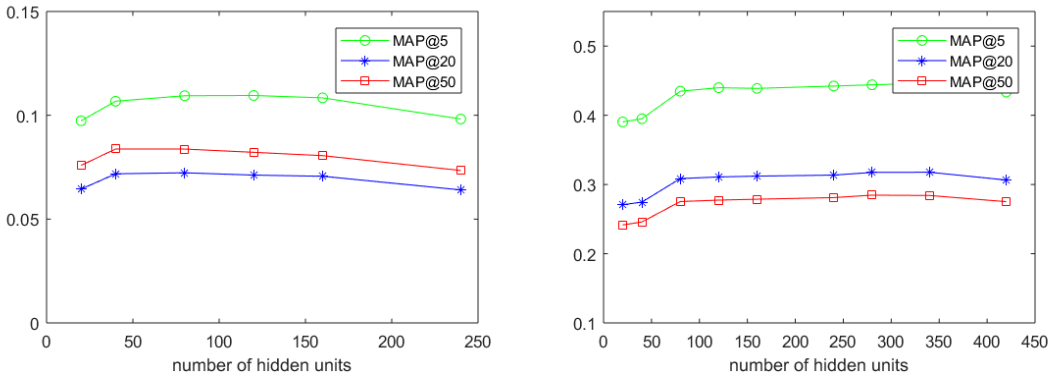


Fig. 2. MAP@k of RsyGAN showing variations in the number of hidden units

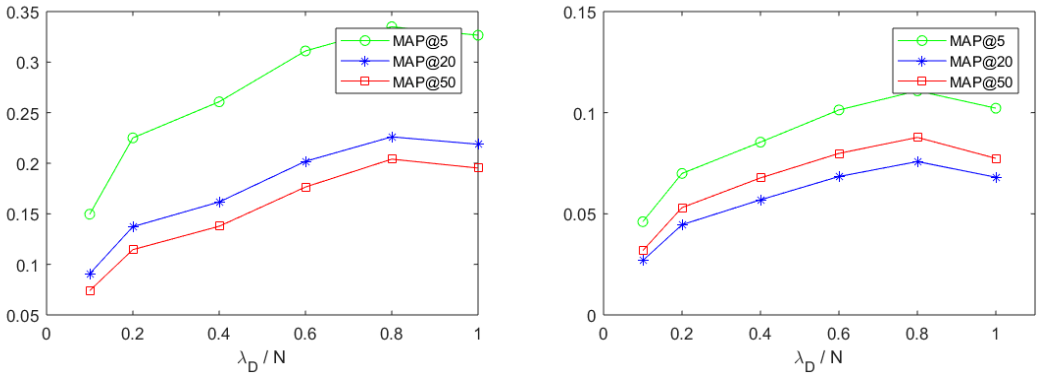


Fig. 3. The effects of parameter $\lambda_D$

ranging from 0 to 1, thus we times N when we apply $\lambda_D$. Figure 3 shows the predictive performance for RsyGAN on the two dataset.

We find that we obtain the best performance on Taobao dataset when $\lambda_D$ equals 0.8. In our extensive experiments, we observe that the value of $\lambda_D$ is same on MovieLens 10M.

Since adversarial training is widely regarded as an effective but unstable technique, we further investigate the learning trend of our proposed method. Figure 4 shows the learning curves of the generative network and the discriminative network on MovieLens 10M dataset. Here we only show the performance measure by the value of the loss function. The results show that while we cannot prove that the loss function will ultimately converge, we can achieve better recommendation performance than other methods.

## V. Conclusions and Further Study

In this paper, we have introduced a novel method for the top-k recommendation task which can be used in a real recommendation scenario with sparse data. The model in our method contains a generative network and a discriminative network. We utilized the adversarial strategy to train this model. The adversarial training framework takes advantage of both networks: the generative network is guided by the signals from the discriminative network, and the discriminative network can be enhanced by the generative network. We also conducted experiments on several datasets and compared the results with state-of-the-art methods. Significant performance gains were observed in each set of experiments.

We plan to combine auxiliary information into this model, such as user profile, product brand information, and product images. In future work, we aim to study how to make the adversarial training process more effective and stable.

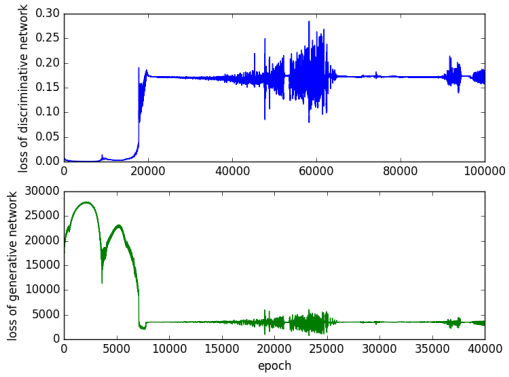


Fig. 4. Learning curves of RsyGAN on MovieLens 10M

## ACKNOWLEDGMENT

This research was supported by National Key R&D Program of China (No.2016YFB0801100), Beijing Natural Science Foundation (No.4172054, No.L181010), and National Basic Research Program of China (No.2013CB329605). This work was also supported by the Australian Research Council (ARC) under Grant [DP170101632].

## REFERENCES

[1] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[2] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrezz, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[3] K. Georgiev and P. Nakov, "A non-IID frameworkfor collaborative filtering with restricted boltzmann machines," in *ICML*, vol. 28, 2013, pp. 1–9.

[4] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 4:2—-4:2, 2009.

[5] Y. Koren, P. Ave, F. Park, H. D. Management, and D. Applications, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *SIGKDD*, 2008, pp. 426–434.

[6] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec : Autoencoders meet collaborative filtering," in *WWW*, 2015, pp. 111–112.

[7] H. Wang, B. Chen, and W. J. Li, "Collaborative topic regression with social regularization for tag recommendation," in *IJCAI*, 2013, pp. 2719–2725.

[8] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *SIGKDD*, 2015, pp. 1235–1244.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *NIPS*, 2014, pp. 1–9.

[10] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, pp. 19:1—-19:19, 2015.

[11] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *WSDM*, 2014, pp. 273–282.

[12] W. Wang, G. Zhang, and J. Lu, "Member contribution-based group recommender system," *Decision Support Systems*, vol. 87, pp. 80–93, 2015.

[13] Y. Zhang, L. Pang, L. Shi, and B. Wang, "Large scale purchase prediction with historical user actions on B2C online retail platform," in *RecSys*, 2014, pp. 5–8.

[14] J. Breese and D. Heckerman, "Empirical analysis of predictive algorithms for collaborative filtering," in *UAI*, 1998, pp. 43–52.

[15] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *ICDM*, 2008, pp. 502–511.

[16] D. Wu, G. Zhang, and J. Lu, "A fuzzy preference tree-based recommender system for personalized business-to-business e-services," *IEEE Transactions on Fuzzy Systems*, vol. 23, pp. 29–43, 2015.

[17] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, "Factorization meets the item embedding: regularizing matrix factorization with item co-occurrance," in *RecSys*, 2016, pp. 59–66.

[18] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.

[19] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 42–49, 2009.

[20] P. Symeonidis, "Matrix and tensor decomposition in recommender systems," in *RecSys*, 2016, pp. 429–430.

[21] F. Yuan, G. Guo, J. M. Jose, and L. Chen, "LambdaFM : Learning optimal ranking with factorization machines using lambda surrogates," in *CIKM*, 2016, pp. 227–236.

[22] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, "A deep matrix factorization method for learning attribute representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 417–429, 2017.

[23] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *ICML*, 2007, pp. 791–798.

[24] F. Strub, J. Mary, and R. Gaudel, "Hybrid collaborative filtering with autoencoders," *arXiv*, 2016.

[25] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for Top-N recommender systems," in *WSDM*, 2016, pp. 153–162.

[26] H. J. Xue, X. Y. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *IJCAI*, 2017, pp. 3203–3209.

[27] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv*, 2017.

[28] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *CEUR Workshop Proceedings*, vol. 1691, 2017, pp. 64–66.