Noname manuscript No.
(will be inserted by the editor)

Adversarial Dual Autoencoders for Trust-aware Recommendation

Manqing Dong* · **Lina Yao** · **Xianzhi Wang** · **Xiwei Xu** · **Liming Zhu**

Received: date / Accepted: date

Abstract Recommender systems face longstanding challenges in gaining users' trust due to the unreliable information caused by profile injection or human misbehavior. Traditional solutions to those challenges focus on leveraging users' social relationships for inferring the user preference, i.e. recommending items according to the preference by user's trusted friends; or adding random noise to the input to improve the robustness of the recommender systems. However, such approaches cannot defend the real-world noises like fake ratings. The recommender model is generally built upon all the user-item interactions, which incorporates the information from fake ratings or spammer groups, that neglects the reliability of the ratings. To address the above challenges, we propose an adversarial training approach in this work. In details, our approach includes two components: a *predictor* that infers the user preference; and a *discriminator* that enforces cohort rating patterns. In particular, the predictor applies an encoder-decoder structure to learn the shared latent information from sparse users' ratings and trust relationships; the discriminator enforces the predictor to provide ratings as coherent with the cohort rating patterns. Our extensive experiments on three real-world datasets show the advantages of our approach over several competitive baselines.

* Corresponding Author
M. Dong · L. Yao.
University of New South Wales. Australia.
E-mail: {manqing.dong, lina.yao}@unsw.edu.au

X. Wang
University of Technology Sydney. Australia.
E-mail: xianzhi.wang@uts.edu.au

X. Xu · L. Zhu
Data 61, CSIRO. Australia.
E-mail: {xiwei.xu, liming.zhu}@data61.csiro.au

Keywords Recommender systems · Autoencoder · Trust · Adversarial training

1 Introduction

Recommender systems offer an effective way of delivering information, products, and services to users with personalized information, which have been proven successful in various domains such as online entertainment and e-commerce [29]. However, users may not trust the recommender systems due to inaccurate recommendation results. For example, a user may not trust a stranger's preference even when they have similar rating records. Another example is that the system may recommend an item that is deliberately highly rated by malicious users.

One traditional solution to the above issues is leveraging external trust relationships, which is often called trust-aware recommendation [19]. Related research diverges into memory-based and model-based methods. The former mainly employ memory-based collaborative filtering methods—they search the trust networks to obtain the neighbors and then make recommendations based on those trusted neighbors [17]. For example, Jamali and Ester [12] combine TrustWalker [11] with neighborhood collaborative filtering. They first use random walks to get the user representation from the trust network and then perform a probabilistic strategy for selecting items to give recommendation. Similarly, Zhang et al. [31] retrieve the user trust information from user feedback and infer the user preference from the top-k identified friends. Model-based methods are majorly apply model-based collaborative filtering methods, such as matrix factorization [28, 7], for recommendation. For example, Zhao et al. [32] incorpo-

rate the social trust information based on a Bayesian Personalized Ranking approach. They assume that the user preference will be affected by their friends, i.e. the user will also leave high ratings to items preferred by their friends. Guo et al. [7] integrates the social trust information with using a SVD++[14] based method. Both memory-based and model based trust-aware recommendation methods improve the model performance by leveraging the explicit or implicit relationship among users. However, they may fail to consider the reliability of ratings in determining the trustworthiness of recommender systems.

Another direction towards trust-aware recommendation is to design a robust recommender system that resists biased or randomized ratings provided by users in a real-world context. One approach is to insert man-made noise into the input to force the system to learn robust parameters of the input so that to improve the model's ability in resisting the noise. One example is the denoising auto-encoder (DAE) [3], which corrupts the inputs with man-made noises. The work [27] used collaborative denoising auto-encoder (CDAE) which shares similar ideas of DAE. The inputs (ratings) are corrupted by the Gaussian noises and then fed into the neural nets via an encoder to get a dense representation. The decoder is trying to map the dense representation into the user-item interactions and for recommendation. Instead of man-made noise, some work adds adversarial noise to the model. The majority of this type of work focus on introducing noise in model configurations to improve the robustness of the model parameters. For example, He et al. [9] introduced an additional objective function in the traditional Bayesian Personalized Ranking approach to quantify the loss of a model under perturbations on its parameters. In details, the adversarial noises are added to the model parameters; the recommender model is updated by considering both the training loss and the adversarial loss, where they minimize the training loss while maximize the adversarial loss. Yuan et al. [30] mixed adversarial noise with model parameters and latent user representations to improve the robustness of the model. Their training strategy includes two learning steps: first, obtain optimal parameters by a training step; and second, minimize the recommendation loss while maximize the adversarial noise loss. Similar to trust relationship-aware recommendation approaches, a limitation of the above proposed noises is that the model cannot defend the real-world noises like fake ratings.

To the best of our knowledge, few studies have focused on the robustness issue caused by user misbehaviors in rating. In this regard, we embrace the advantages of adversarial training in simulating biased or malicious ratings and propose reinforced trust-aware recommendation to harvest the benefits of both social information and the denoising approach. Our method consists of a predictor that infers the ratings and a discriminator that enforces cohort rating patterns on the predicted ratings. In a nutshell, we make the following contributions:

- We propose a rating predictor based on an encoder-decoder structure to learn latent information about user rating patterns and user social trust networks. User social trust embedding learned by an attentive graph neural network can balance the contributions of user neighbors. The predictor distinguishes from previous studies in considering not only user's trust relationship but also rating quality.
- We introduce a discriminator to learn transferable patterns in rating behaviors while eliminating user-specific bias, thereby enforcing consistent rating patterns among different users to lift the robustness of the model.
- We have tested the proposed model on three real-world datasets to show its competitive performance against several baselines. We provide detailed parameter studies and model discussions.

We will review the related work for social-aware recommendation and the robustness of the recommender systems in the following section 2. We further present our proposed method in section 3 and show the model performance in section 4. The conclusion and future works are discussed in section 5.

2 Related work

Traditional recommendation techniques to deal with the trust issue include the exploitation of social relations or adding randomly generalized noise to the model configurations to improve the robustness of the recommender system [2].

Social-aware recommendation approaches utilize the user trust network to complement the sparse rating data. This will improve the recommendation performance by considering two source of rating information: the original user preference and the preference from the trusted users. Traditional social-aware recommendations include memory-based methods and model-based methods. The former mainly propose trust propagation methods by leveraging the ratings of friends to deduce the ratings of a targeted user [26]. The work by [18] is one of the first works that leverages the social relationships. The idea is replacing the role of collaborative filtering by trust network. Specifically, the model

propagates trust information over the social trust network to estimate the weight for the trust link that can be used in place of the user similarity weight. Jamali and Ester [12] combine TrustWalker [11] with neighborhood collaborative filtering. They first use random walks to get the user representation from the trust network and then perform a probabilistic strategy for selecting items to give recommendation. Zhang et al. [31] retrieve the user trust information from user feedback and infer the user preference from the top-k identified friends. Model-based methods largely depend on matrix factorization. The social relations are generally used to form the user representation. For example, Wen et al. [25] use graph embedding approaches for learning learn the user social trust representation and then combine the trust representation with user ratings as the input of matrix factorization. Guo et al. [7] integrates the social trust information with using a SVD++[14] based method. Ahn et al. [1] have quantified by how much social network information can reduce sample complexity, which provides the theoretical support for integrating the social trust information. Zhao et al. [32] incorporate the social trust information based on a Bayesian Personalized Ranking approach. They assume that the user preference will be affected by their friends, i.e. the user will also leave high ratings to items preferred by their friends.

Another direction is designing robust recommender systems. A general way is introducing noise to the system configurations to improve system performance. By doing so, the model is forced to learn robust parameters to improve denoising capability. Traditional methods include introducing human-made noise. For example, in the collaborative denoising auto-encoder [27], the input data are corrupted by Gaussian noise before fed to the neural network. The decoder is trying to map the dense representation into the user-item interactions and thus for recommendation. Wang et al.[23] integrate both recurrent neural networks (RNNs) [15] and denoising autoencoders for recommendation. The RNNs are used for extracting the information from the item textual description. The whole model is in an autoencoder structure, where the RNNs are used as encoder and decoder layers. The proposed recurrent autoencoder can learn both rating information and sequential information (e.g. textual information) to get the dense representation. Strub et al. [21] corrupt the inputs by stacked denoising autoencoders[22]. They also considered the side information, e.g. user profiles and item profiles, to enhance the robustness of the model. In the later researches, some works leverage adversarial noise to improve the robustness of the model. Wang et al. [24] propose a generative adversarial model that consists of a generator and a discriminator for recommendation [6]. The generator (predictor) acts as an attacker to cheat the discriminator by capturing the rating patterns from the users and generating ratings with similar patterns; the discriminator targets distinguishing the generated samples from the real ratings. The two models update step by step by competing with each other, like playing a minimax game until the generator (predictor) provides well and stable rating prediction. He et al. [9] introduced an additional objective function in the traditional Bayesian Personalized Ranking approach to quantify the loss of a model under perturbations on its parameters. In details, the adversarial noises are added to the model parameters; the recommender model is updated by considering both the training loss and the adversarial loss, where they minimize the training loss while maximize the adversarial loss. Yuan et al. [30] mixed adversarial noise with model parameters and latent user representations to improve the robustness of the model. Their training strategy includes two learning steps: first, obtain optimal parameters by a training step; and second, minimize the recommendation loss while maximize the adversarial noise loss.

However, The above two directions of trust-aware recommender systems do not consider the reliability of the ratings, i.e., the existence of biased, randomized, or malicious ratings provided by users. The former social-aware approaches mostly do not consider the robustness issues, and the denoising approaches majorly focus on the parameter robustness. In this paper, we bridge the advantages of both social-aware recommender systems and robustness issues for the recommendation with reinforcing cohort rating patterns.

3 Methodology

3.1 Overview

In this work, we consider the rating prediction problem in recommender systems. Our target is predicting users' ratings on new items based on the user-item rating interactions and social trust relationships. Let $R \in \mathbb{R}^{m \times n}$ denotes the user-item rating matrix, where each entry $r_{u,i}$ represents the rating of user u on item i; m and n are the numbers of users and items, respectively. We use I_u to represent the set of items rated by user u and r_u to represent the according ratings. The social network can be represented by a graph $G = (V, E)$, where V is a set of m nodes (users), and E denotes directed trust relations among users. We use T to describe the weight of E, where $t_{u,v} \in T$ indicates the trust degree between u and v. The trusted users by user u is represented by V_u, i.e. $\{t_{u,v} = 1 | v \in V_u\}$. The ratings from the

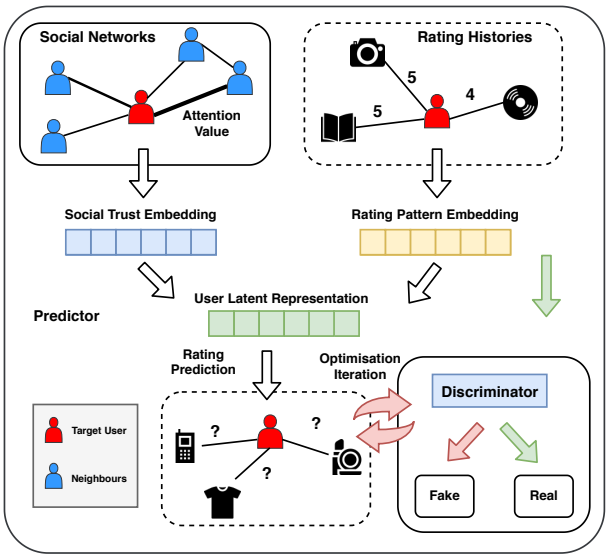


Fig. 1 Structure of our proposed end-to-end model. The predictor predicts users' ratings based on their previous ratings and trust relationships. The discriminator enforces consistent predictions regardless of individuals' behavioral differences in rating.

trusted users are denoted as $\{r_v|v \in V_u\}$. Thus the recommender model is trying to predict $r_{u,i}$ for new items by $r_{u,i} \leftarrow (r_u, r_{V_u})$. Figure 1 illustrates the structure of the proposed model, where we have the recommender model works as the predictor and a discriminator to force the cohort rating patterns in the predicted ratings. The predictor first learns the latent representation of users' ratings and trust relations and then combines them into a shared hidden layer that contains users' latent patterns. It also acts as a generator to simulate rating patterns of real users. The rating pattern embedding is learned from neural networks, i.e. $H_r \leftarrow r_u$; while the social trust embedding is learned by attentive graph neural networks [5], i.e. $H_t \leftarrow \{r_v|v \in V_u\}$. The discriminator determines whether the predicted ratings $\{\hat{r}_{u,i}|i \in I_u\}$ follow the cohort patterns as the meta-information $\{r_{u,i}|i \in I_u\}$, thereby providing accurate and confidential rating prediction. It also detects the abnormal rating patterns to improve the robustness of the model. We provide more details about the proposed model as in the followings.

3.2 Rating Prediction with Correlative Trust Relationship Fusion

Autoencoder is an unsupervised model that reconstructs its inputs in the output layer, which has been used in many recommendation tasks [20]. The encoder-decoder structure can help with learning the latent preferences of users according to the user-item interactions and providing predictions based on the latent preferences. In this work, we integrate trust information into the layers to conduct comprehensive recommendations. We first learn a shared latent representation from two types of sparse information: users' previous ratings and ratings from trust users, i.e. dual autoencoders, and then predict ratings based on that representation.

3.2.1 Embedding Learning

Here we learn two types of sparse information to get the latent representation, i.e. social trust embedding and rating pattern embedding.

The meta representation for the user rating pattern is simply represented by r_u. To infer the rating pattern embedding, the encoder layer maps the inputs into a low-dimensional space by neural networks. The simplest case is using fully connected layers:

$$H_r = \sigma(W_e^\top r_u + b_e^r) \tag{1}$$

where W_e is the weight in encoder layers, b_e^r is the bias term, and σ is the activation function. The encoder layer could also be in other forms, such as convolutional neural network [15], according to the learning tasks.

The meta representation of the social trust relationships is learned from the rating patterns of the trusted users. Given a set of trusted users V_u, i.e., $t_{u,v} = 1$ for $v \in V_u$, where each user has a rating record r_v, we employ an attentive graph neural network [5] for learning the social trust meta representation s_u:

$$s_u = \sigma(W_s^\top \Sigma_{v \in V_u} \alpha_v r_v + b_s) \tag{2}$$

where α represents the contribution of user v, which could be regarded as attention values; σ stands for the activation function; W_s and b_s are weights and biases. We will hereby omit explanation of similar notations of weights and bias for simplicity. Intuitively, the neighbors of user u are not equally contributed to the social trust representation of users; thus, we utilize the attention mechanism, i.e. the attention values α_v proposed in equation 2, to balance the social influences. Suppose user u has strong connections with the neighbors who has similar tastes, then we learn the attention value for each user as follows:

$$\beta_v = \sigma(W_a^\top f(r_u, r_v) + b_a) \tag{3}$$

$$\alpha_v = \frac{\exp \beta_v^\top w_v}{\Sigma_{v \in V_u} \exp \beta_v^\top w_v} \tag{4}$$

where $f(r_u, r_v)$ is the correlation function representing the correlative rating patterns between user u and trusted user v; w_v is a randomly initialized vector that captures the correlative latent patterns. The correlation

function evaluates the rating pattern similarity between the user and the trusted users. It can be in different forms. For example, the correlation function can be the concatenation or the difference of two rating lists. We will discuss the model performance on different correlation functions in the further experiments. Now we have the social trust meta representation s_u. Similarly, the encoder layer will map the meta representation s_u into a low-dimensional space by neural networks:

$$H_t = \sigma(V_e^\top s_u + b_e^s) \tag{5}$$

where V_e stands for the weights.

The two encoder layers works simultaneously. Given the user history ratings r_u and the information of the trusted users, we get the social trust embedding H_t and rating pattern embedding H_r by several encoder layers.

3.2.2 Rating Prediction

After several encoder layers, we get more concise representations of the rating records as H_r and trust information as H_s.

To integrate two sources of information, we sum up the latent representations H_r and H_s with weights to form a shared latent representation:

$$H = \gamma \cdot H_r + (1 - \gamma) \cdot H_t \tag{6}$$

where $\gamma \in [0, 1]$ is the parameter to control the contribution of the rating information to the shared latent representation in comparison with social trust information. Another way to combine two sources of information is concatenating the latent representation, i.e., $H = [H_r, H_t]$. The experimental results showed that the summing of the two representations performs better than the concatenation of the two representations. We will discuss the performances of such two ways later in the ablation studies.

Differing from the encoder, the decoder aims to explain or expand the concise latent representation. Given the concentrated information about a user's preferences embedded in the shared latent representation, we obtain the predicted ratings $\hat{r}_u$ by decoding it into a list of ratings and trust relationship:

$$\hat{r}_u = \sigma(W_d^\top H + b_d). \tag{7}$$

The performance of the recommendation can be evaluated by the loss between the original inputs and the predictions, i.e., $\ell(r, \hat{r})$ and $\ell(t, \hat{t})$, where ℓ is the loss function.

3.3 Cohort Rating Patterns Enforcement

The predictor works fine alone after training but may neglect noises in the input, due to the possible diverse rating distributions from abnormal users in a real-world context. We design a discriminator to distinguish the generated ratings $\hat{r}$ from real ratings and train the model until the discriminator cannot classify them accurately [6]. This way, we can enforce cohort rating patterns on the generated ratings to reduce the adverse impact of users' biases, misbehaviors, and low-quality ratings. We use a multilayer perceptron as the classifier to predict any type of rating inputs (r_u or $\hat{r}_u$), say r_*:

$$\hat{y} = D(r_*) = \text{softmax}(\sigma(W_c^T r_* + b_c)) \tag{8}$$

We train the classifier in two steps: discriminating and generating. In the first step, a discriminator aims to output $y = 0$ for any generated rating $\hat{r}$ and $y = 1$ for real ratings r, by minimizing

$$L_D = E_{r_* \in \{r, \hat{r}\}}[\ell(y, \hat{y})] + \lambda ||\Theta||_1 \tag{9}$$

via gradient descent, where $E_{r_* \in \{r, \hat{r}\}}[\ell(y, \hat{y})]$ is the mean prediction loss for our generated ratings or the real ratings, ℓ is the cross entropy loss function, Θ represents model parameters, λ is the hyper-parameter, and $||\Theta||_1$ is the regularization item to avoid over-fitting, where here we use absolute-value norm for regularization. Since the generated ratings $\hat{r}$ are not as sparse as the real data (the real data is sparse due to the limited user-item rating records), we multiply them with a mask vector before feeding them into the discriminator, where the i-th element will be zero if a user does not provide a rating to item i.

The generating step trains a predictor to cheat the discriminator the discriminator aims to output $y = 1$ for the generated ratings ($\hat{r}$) by minimizing the gaps between the predicted labels of generated ratings $\hat{r}$ and $y = 1$, in order to learn a transferable rating patterns. The loss for the generating step is:

$$L_G = E_{r_* \in \hat{r}}[\ell(1, D(r_*))] + \lambda ||\Theta||_1 \tag{10}$$

The whole process iterates until the discriminator cannot predict the generated ratings correctly.

Before training the discriminator, we train the predictor, i.e. our recommender model by minimizing the rating prediction loss (i.e., mean squared loss ℓ) until convergence to train an accurate and robust model:

$$L_R = \ell(r, \hat{r}) + \lambda ||\Theta||_1 \tag{11}$$

The overall training process of our method is described in algorithm 1, where the model is updated for $u \in U^{Train}$. The pseudo code of the testing phase is

Algorithm 1 Training Phase

Input: A set of training users U^{Train}, the user rating records $\{r_u|u \in U^{Train}\}$ and the rating records of their trusted users $\{r_v|v \in V_u\}$

Set initial values for hyper-parameters λ and γ
Randomly initialize weights W_*, V_* and biases b_*
while Not done **do**
  for $u \in U^{Train}$ **do**
    Get rating pattern embedding $H_r \leftarrow r_u$ by equation 1
    Get social trust representation $s_u \leftarrow (r_u, \{r_v|v \in V_u\})$ by equation 2
    Get social trust embedding $H_t \leftarrow s_u$ by equation 5
    Integrate the two representations with $H = \gamma \cdot H_r + (1 - \gamma) \cdot H_t$
    Calculate the prediction $r_{u,i} \leftarrow H$ by equation 7
    Calculate the adversarial loss L_D, L_G, and the recommendation loss L_R by equation 9, 10, 11, respectively.
  end for
  Update the recommender model by minimizing L_R
  Update the model and discriminator by minimizing L_D and L_G
end while

Algorithm 2 Testing Phase

Input: A set of testing users U^{Test}, the user rating records $\{r_u|u \in U^{Test}\}$ and the rating records of their trusted users $\{r_v|v \in V_u\}$

Set values for hyper-parameters λ and γ
Initialize weights W_*, V_* and biases b_* with learned values
for $u \in U^{Test}$ **do**
  Get rating pattern embedding $H_r \leftarrow r_u$ by equation
  Get social trust representation $s_u \leftarrow (r_u, \{r_v|v \in V_u\})$ by equation 2
  Get social trust embedding $H_t \leftarrow s_u$ by equation 5
  Integrate the two representations with $H = \gamma \cdot H_r + (1 - \gamma) \cdot H_t$
  Get the prediction $r_{u,i} \leftarrow H$ by equation 7
end for

showed in algorithm 2. In the actual experimental settings, we update the model with batch of users. Specifically, we update the recommender model and the discriminator asynchronously. The target of our method is constructing an accurate and robust model; the design for the cohort rating patterns enforcement will help the model produce reliable rating predictions. Thus, we update the recommender model with every batch of training users, and we update the discriminator with every few batches of training users.

4 Experiments

4.1 Datasets

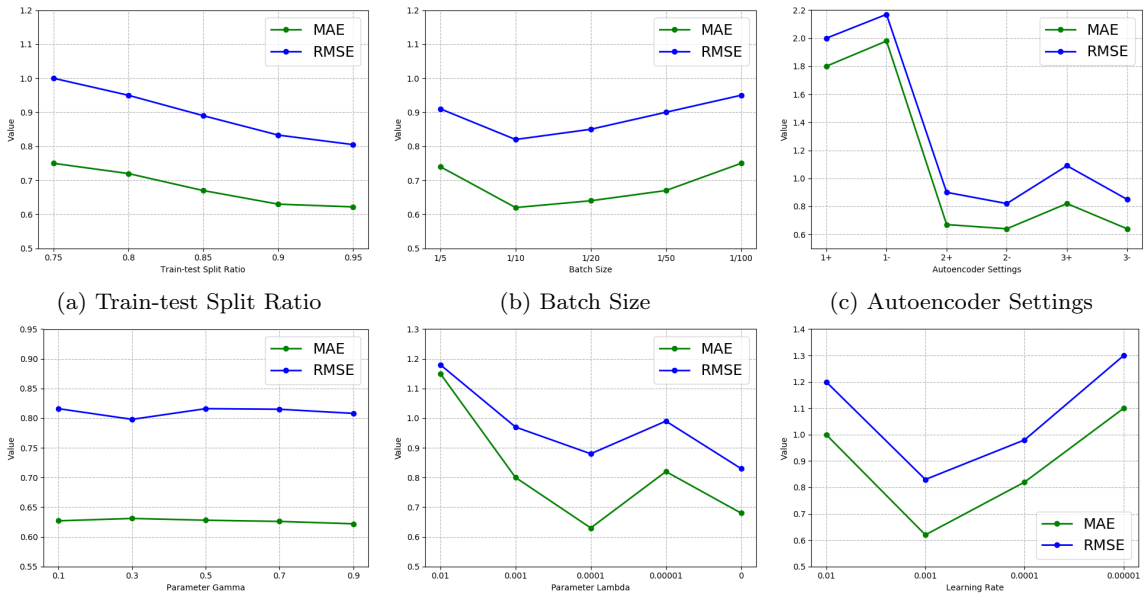
We evaluate the proposed model on three real-world datasets: *FilmTrust*, *Epinions* and *Ciao*[1]. *FilmTrust* is a small dataset that consists of 35,497 ratings of 2,071 items from 1,508 users, and 1,853 trust links. The latter two datasets contain over 100 thousand items from thousands of users. For *Epinions*, which is a product review dataset, there are 469,126 ratings from 37,701 users in 19,627 items. There are about 487,000 trust relationships among users. *Ciao* consists of 137,187 ratings from 7237 users for 8,819 products, and there are 111,781 trust links.

4.2 Model Setups

Data-preprocessing. First, we filter the missing values of the dataset. Second, we preprocess the two larger datasets to make them applicable to our method. The *Epinions* and *Ciao* are two large dataset that contains over 100 thousand items. Our approach is based on the user-item interactions, i.e. the rating records for each user is represented by $r_u \in \mathbb{R}^n$, where n is the number of items. The computation cost of our method will be high if with a large n; besides, using fully connected layers for encoding the inputs will aggravate such situation. There are two ways for alleviating the computation cost: decrease the number of items or use less complicated model structure (less parameters). Thus, we filter the dataset with items that with less than 10 rating records, and we use convolutional layers for encoding the inputs.

Experimental settings. Our code is implemented with TensorFlow [2] in Python 3.7 and runs on a Linux server with NVIDIA TITAN X. The processed datasets will take about 60MB hard disk space. The default activation function is Sigmoid function [8]. We have parameters in model set-ups, encoder-decoder structures, and hyperparameters. By default, we use 90% of each dataset for training and others for testing; the batch size is about 1/10 of the dataset; we use two encoder layers for encoding the inputs and two decoder layers for decoding the latent representation; the hyperparameter γ for controlling the contribution of rating pattern embedding is set as 0.7; the hyperparameter λ, which is the coefficient for the regularization item, is set as 0.001; the learning rate is 0.001.

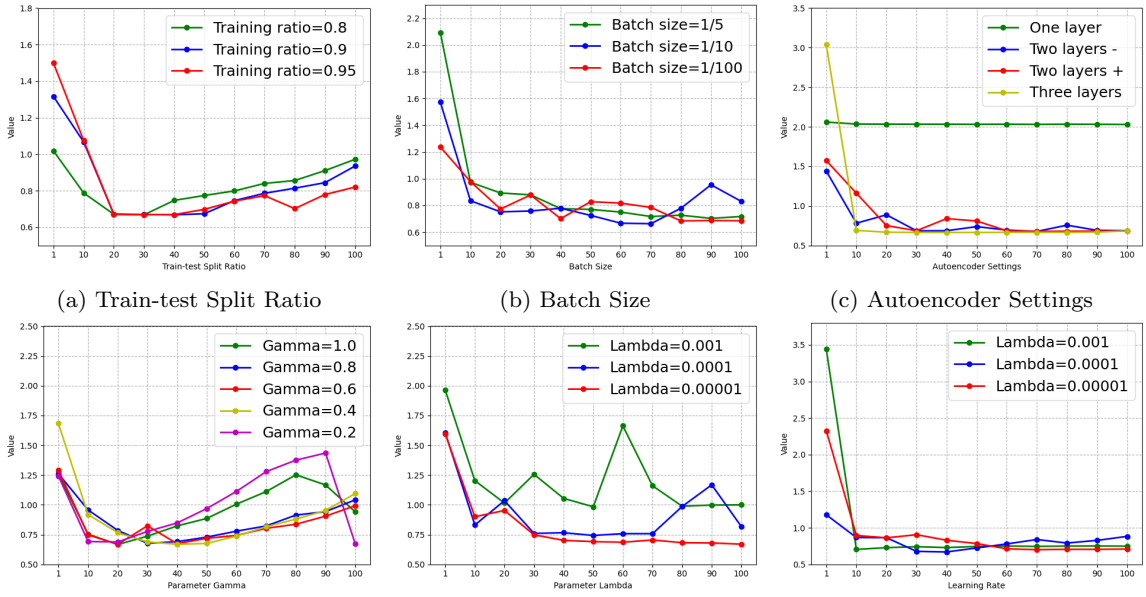
[1] https://www.librec.net/datasets.html
[2] https://www.tensorflow.org/



(a) Train-test Split Ratio (b) Batch Size (c) Autoencoder Settings

(d) Parameter γ (e) Parameter λ (f) Learning Rate

Fig. 2 Sensitivity to parameter settings.



(a) Train-test Split Ratio (b) Batch Size (c) Autoencoder Settings

(d) Parameter γ (e) Parameter λ (f) Learning Rate

Fig. 3 Model performance during the training process.

4.3 Parameter Studies

We have three types of parameters for setting up the model: the data set-up parameters, the encoder-decoder structure settings, and the hyperparameters. In this section, we study on the performance of our proposed model with different settings on the *FilmTrust* dataset. We will show the results under different settings and different learning epochs. The results are under two evaluation metrics: Mean Absolute Error (MAE) and Root-Mean Squared Error (RMSE).

- MAE: $\frac{1}{m}\Sigma_{u=1}^{m}\frac{1}{n}\Sigma_{i=1}^{n}|r_{u,i}-\hat{r}_{u,i}|$
- RMSE: $\sqrt{\frac{1}{m}\Sigma_{u=1}^{m}\frac{1}{n}\Sigma_{i=1}^{n}(r_{u,i}-\hat{r}_{u,i})^2}$

A lower value indicates better model performance.

Data set-up parameters include the train-test split ratio and batch size. Default settings for these parameters are 0.9 (for training dataset), and the batch size is about 1/10 of the training dataset. Figure 2 (a)-

(b) show the overall model performance on different settings. The results suggest a larger training set improves the model performance; and the model with a moderate batch size, rather than the extreme settings of the batch size (e.g., 1/100 or 1/5), delivers the best performance. Figure 3 (a)-(b) show the model performance during the training process. We could see that a larger training set also improves the stability of the model, where the model performs best when training ratio is 0.95. According to figure 3 (b), the model performance fluctuates with a small batch size while converges slowly with a large batch size.

Encoder-decoder structure setting. We compare the models under different settings regarding the number of encoder/decoder layers (1, 2, 3) and the number of neural nodes (1/20 to 1/2 of the dimension of inputs) in the hidden layers. We show results of models with one, two, and three layers, and we use '+', '-' to indicate higher (e.g. 1/10 to 1/20 of the dimension of inputs) or lower dimensions (e.g. 1/2 to 1/10 of the dimension of inputs) of layer nodes. Our experimental results (Figure 2 (c)) reveal that adding more layers to the encoder or the decoder delivers better performance, due to the sparsity and high dimensionality of the datasets. The two-layer structure delivers very similar results as the three-layer structure, though the performance slightly fluctuates for a three-layer structure under high dimensionality. Smaller dimensions of layer nodes generally result in better performance, given the same number of layers (except for one layer). The figure 3 (c) also suggests that it is hard for the model to learn effective patterns with only one neural layer, and a three layers encoder with lower dimensionality provides most stable prediction.

Hyperparameters. Figure 2(d)-(f) show the performance over the hyperparameters (γ, λ) and learning rate. γ controls the weight for user rating patterns in comparison with user social trust embedding. Our experiment on γ reveals that bias in user preference may lead to better performance of our model. Besides, using only one source information (ratings or trust relations) delivers inferior results, indicating there exist hidden relationships between users, rating behaviors, and their trust relationship. According to figure 3 (d), we could also observe that using only one source information will aggravate the over-fitting issue. So a median value of γ provides better and stable performance for the recommendation. λ is the regularization coefficient. According to both figure 2 (e) and 3 (e), a small value of λ (between 0.0001 and 0.00001) provides best and reliable results, while a larger value (e.g. over 0.001) or a near zero value will lead to a bad model performance. As for the learning rate, it is reasonable to set the learning rate to a moderate value because large values tend to make the convergence difficult while smaller values may slow down the learning. Here, the value 0.001 provides the best performance.

4.4 Comparison Results

We compare the proposed model with several baseline algorithms, including TrustMF [28], SoReg [16] and SocialMF [13]. These methods use matrix-factorization based methods and combining social information into user embedding. Besides, considering the popularity of deep learning in the recent recommendation research, we compare three recent deep learning-based methods for comparison, which are NeuMF [10], DeepSoR [4], and GraphRec [5].

- TrustMF: constructs a trust network and maps the users into truster space and trustee space. Each user has feature vectors in the trust networks, and the representation for each user is affected by the trusted users. Then collaborative filtering method is used for recommendation.
- SoReg: employs social trust networks to get regularization terms for controlling the matrix factorization objective function.
- SocialMF: also considers the matrix factorization methods, where they incorporates the user social information for forming the user representation. The prediction is based on the user representation and item representation.
- NeuMF: is a matrix factorization model with neural network architecture.
- DeepSoR: forms user representation from social networks and use probabilistic matrix factorization for rating prediction.
- GraphRec: models two graphs, i.e., the user-user social graph and the user-item graph, with graph neural networks; the rating prediction is based on the concatenation of item representation and user representation.

The above comparison methods all consider the social relationships for recommendation, while they are mostly based on matrix factorization algorithms. The GraphRec method is similar to our work that they use graph neural networks to infer the user trust embedding from the social relationships; but it ignores the reliability of the user ratings. We reuse the default parameters or the presented results from the original papers for comparison. Table 1 shows the experimental results, where the last row stands for the performance of our model. We can see that both matrix-factorization

Table 1 Comparison Results

Methods	FilmTrust		Ciao		Epinions	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
TrustMF	0.631	0.810	0.769	1.048	0.939	1.167
SoReg	0.668	0.875	0.861	1.085	0.932	1.232
SocialMF	0.638	0.837	0.827	1.050	0.825	1.070
NeuMF	0.655	0.867	0.806	1.062	0.907	1.148
DeepSoR	0.648	0.853	0.774	1.032	0.838	1.097
GraphRec	0.633	0.819	0.739	0.939	0.817	1.063
Ours	0.622	0.805	0.748	0.976	0.815	1.054

based methods and deep learning-based methods use users' latent preferences for recommendation while in different ways. Both methods achieved similar performance on the small *FilmTrust* dataset. Deep learning-based methods perform better on the *Ciao* and *Epinions* datasets, which are much larger than *FilmTrust*. This can be attributed to the stronger capability of deep neural networks in capturing complex relationships among input. Due to the ascendant ability in handling complex graph structures, GraphRec performs better than our model in dataset *Ciao*, which has a higher density of social links and user-item ratings. Overall, our model performs consistently well on three datasets and outperforms a series of comparison methods, which shows the effectiveness of our proposed attentive graphical user trust relationship learning and the adversarial training strategy. We will discuss the details about these modules in the following sections.

4.5 Ablation Studies

In this section, we carry out a series of ablation studies to show the effectiveness of leveraging both of the trust information and the robustness of recommender systems, i.e. our attentive graphical learning for user trust representation and the adversarial training strategy for updating the recommender model. Besides, we discuss the model performance with different settings of the correlation function, which is designed to balance the neighbors' contributions to the user social trust representation. We perform the studies on the *FilmTrust* dataset.

4.5.1 Impact of Social Trust Information

We tested two methods to combine the latent representations of the rating and trust information. The first method concatenates representations of rating and trust data for each user; the second sums up the representations with different weight settings (as introduced in our method). According to the results listed in Table 2, the second method exhibited better prediction performance in our experiments.

Table 2 Impact of Social Trust Information

Settings	MAE	RMSE
Concatenation	0.722	0.965
Sum, $\gamma = 0$ (without rating info)	0.627	0.816
Sum, $\gamma = 0.3$	0.631	0.798
Sum, $\gamma = 0.5$	0.628	0.816
Sum, $\gamma = 0.7$	0.626	0.815
Sum, $\gamma = 1$ (without social info)	0.635	0.822

The reason may lie in that our designs of user social trust embedding share similar data structures with user rating pattern embedding. The sum up way keeps more structural information than simply concatenation. Besides, we tested the model performance with different settings of parameter γ, which controls the weights for user rating pattern embedding when summed up with user social trust embedding. $\gamma = 0$ and $\gamma = 1$ indicate the cases that model is trained without and solely based on user rating history, respectively. We could observe that the combination of user social trust embedding and the rating pattern embedding performs better than a single perspective of embedding. Interestingly, the model performs well with only the user social trust embedding, confirming our assumption that users share similar tastes with their neighbors.

4.5.2 Impact of Adversarial Training

To validate the discriminator's effectiveness in enforce cohort rating patterns among the real and generated ratings, we compare the ratings generated by our proposed model and those generated solely by the predictor (without adversarial training). Figure 4 shows the model performance and the distribution of the ratings. First, we can see that the model with adversarial training consistently outperforms the model with solely the predictor. Second, compared with ratings generated by the sole predictor, the predicted ratings with adversar-

ial training tend to fall into different ranges for different items with similar patterns as real ratings.

4.5.3 Impact of Correlation Function

The correlation function $f(r_u, r_v)$ defines the relationship between a user and its neighbors to learn the neighbors' contributions to the user's social trust embedding. Intuitively, users with high consistency in history rating records may share similar tastes. We compare the following correlation functions:

- Cosine similarity: $f(r_u, r_v) = \frac{r_u \cdot r_v}{||r_u|| ||r_v||}$
- Concatenation: $f(r_u, r_v) = [r_u, r_v]$
- Difference: $f(r_u, r_v) = r_u - r_v$
- Dot product: $f(r_u, r_v) = r_u \cdot r_v$
- Equal contribution: $f(r_u, r_v) = 1$

Figure 5(a) gives an example of the user social trust representation learning with cosine similarity. The cosine similarity is calculated based on the rating history of user and her neighbors. We further use the cosine similarity to learn the contribution of $r_{\text{Friend 1}}$ and $r_{\text{Friend 2}}$ to the social trust embedding s_u with attention mechanism, referring to equation (2) to (4). Figure 5 (b) shows the results with different settings. The performance of the equal contribution measure indicates friends are not equally contributed to the user social trust representation. Instead, other measures that consider the distance between the user and her neighbors present well performance. However, the performance of the difference measure is quite unstable; it may eliminate the information when users have the same ratings on the same items.

5 Conclusion

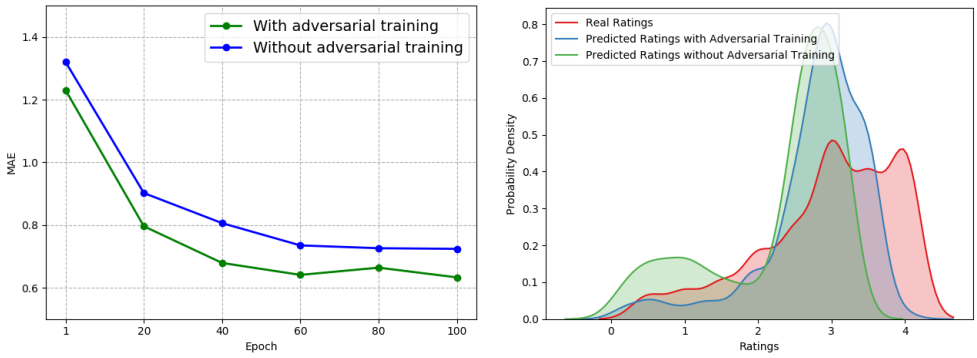
In this work, we propose a unified reinforced trust-aware recommendation model that leverages both users' trust relationships and rating quality to improve the recommendation performance. The model employs a predictor based on an encoder-decoder structure to learn the shared latent information from sparse user ratings and trust relationships, and a discriminator to enforces cohort rating patterns on the predicted ratings. We compare the proposed method with a series of baselines and state-of-arts, and discuss the model performance under different configuration. The experiments on three datasets show the model's competitive performance. One limitation of our proposed method is that the computation cost would be high with larger number of items. We will address this issue in the future work.

Conflict of interest

University of New South Wales; University of Technology Sydney; Data61, CSIRO

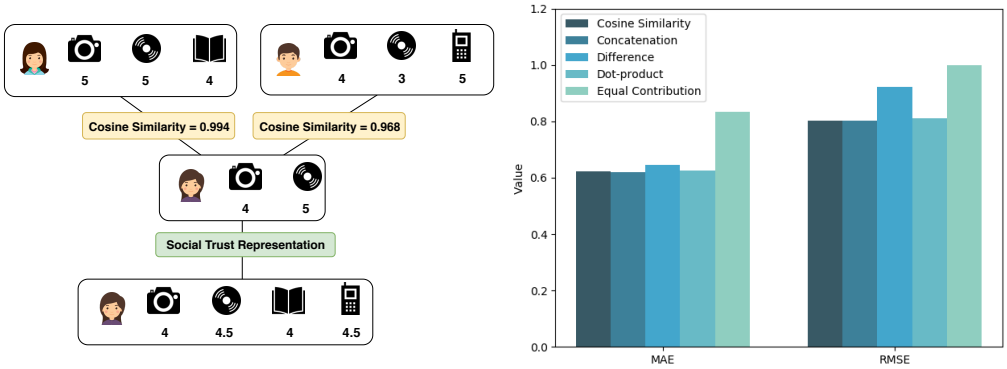
References

1. Ahn, K., Lee, K., Cha, H., Suh, C.: Binary rating estimation with graph side information. In: Advances in Neural Information Processing Systems, pp. 4272–4283 (2018)
2. Dong, M., Yuan, F., Yao, L., Wang, X., Xu, X., Zhu, L.: Trust in recommender systems: A deep learning perspective. arXiv preprint arXiv:2004.03774 (2020)
3. Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., Zhang, F.: A hybrid collaborative filtering model with deep structure for recommender systems. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
4. Fan, W., Li, Q., Cheng, M.: Deep modeling of social relations for recommendation. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
5. Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: The World Wide Web Conference, pp. 417–426. ACM (2019)
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
7. Guo, G., Zhang, J., Yorke-Smith, N.: Trustsvd: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI (2015)
8. Han, J., Moraga, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: International Workshop on Artificial Neural Networks, pp. 195–201. Springer (1995)
9. He, X., He, Z., Du, X., Chua, T.S.: Adversarial personalized ranking for recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 355–364. ACM (2018)
10. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp. 173–182. International World Wide Web Conferences Steering Committee (2017)
11. Jamali, M., Ester, M.: Trustwalker: a random walk model for combining trust-based and item-based recommendation. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 397–406. ACM (2009)
12. Jamali, M., Ester, M.: Using a trust network to improve top-n recommendation. In: Proceedings of the third ACM conference on Recommender systems, pp. 181–188. ACM (2009)
13. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: Proceedings of the fourth ACM conference on Recommender systems, pp. 135–142. ACM (2010)
14. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 426–434. ACM (2008)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)



(a) (b)

Fig. 4 (a) The performance on model with/without adversarial training. (b) The distribution of ratings. Red area shows the distribution for real ratings; green area shows the predicted ratings without adversarial training; and the blue area shows the predicted ratings with adversarial training.



(a) (b)

Fig. 5 (a) An example for the user social trust representation learning with correlation function as cosine similarity. (b) The model performance with different settings of correlation function.

16. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the fourth ACM international conference on Web search and data mining, pp. 287–296. ACM (2011)
17. Ma, X., Lu, H., Gan, Z., Zeng, J.: An explicit trust and distrust clustering based collaborative filtering recommendation approach. Electronic Commerce Research and Applications **25**, 29–39 (2017)
18. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proceedings of the 2007 ACM conference on Recommender systems, pp. 17–24. ACM (2007)
19. Pan, Y., He, F., Yu, H.: Trust-aware collaborative denoising auto-encoder for top-n recommendation. arXiv preprint arXiv:1703.01760 (2017)
20. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, pp. 111–112. ACM (2015)
21. Strub, F., Gaudel, R., Mary, J.: Hybrid recommender system based on autoencoders. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 11–16. ACM (2016)
22. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research **11**(Dec), 3371–3408 (2010)
23. Wang, H., Xingjian, S., Yeung, D.Y.: Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In: Advances in Neural Information Processing Systems, pp. 415–423 (2016)
24. Wang, Z., Gao, M., Wang, X., Yu, J., Wen, J., Xiong, Q.: A minimax game for generative and discriminative sample models for recommendation. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 420–431. Springer (2019)
25. Wen, Y., Guo, L., Chen, Z., Ma, J.: Network embedding based recommendation method in social networks. In: Companion of the The Web Conference 2018 on The Web Conference 2018, pp. 11–12. International World Wide Web Conferences Steering Committee (2018)
26. Wu, Q., Zhang, H., Gao, X., He, P., Weng, P., Gao, H., Chen, G.: Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. arXiv preprint arXiv:1903.10433 (2019)
27. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 153–162. ACM (2016)
28. Yang, B., Lei, Y., Liu, J., Li, W.: Social collaborative filtering by trust. IEEE transactions on pattern analysis and machine intelligence **39**(8), 1633–1647 (2017)

29. Yu, Y., Gao, Y., Wang, H., Wang, R.: Joint user knowledge and matrix factorization for recommender systems. World Wide Web **21**(4), 1141–1163 (2018)
30. Yuan, F., Yao, L., Benatallah, B.: Adversarial collaborative neural network for robust recommendation. In: Proceedings of the 42th International ACM SIGIR conference on Research and Development in Information Retrieval (2019)
31. Zhang, C., Yu, L., Wang, Y., Shah, C., Zhang, X.: Collaborative user network embedding for social recommender systems. In: 17th SIAM International Conference on Data Mining, SDM 2017, pp. 381–389. Society for Industrial and Applied Mathematics Publications (2017)
32. Zhao, Z., Yang, Q., Lu, H., Weninger, T., Cai, D., He, X., Zhuang, Y.: Social-aware movie recommendation via multimodal network learning. IEEE Transactions on Multimedia **20**(2), 430–440 (2017)