Attentive Social Recommendation: Towards User And Item Diversities

Dongsheng Luo[1]   **Yuchen Bian**[2]   **Xiang Zhang**[1]   **Jun Huan**[3]

[1]The Pennsylvania State University, [2]Baidu Research, USA, [3]StylingAI Inc.
[1]{dul262,xzz89}@psu.edu, [2]yuchenbian@baidu.com, [3]lukehuan@shenshangtech.com

Abstract

Social recommendation system is to predict unobserved user-item rating values by taking advantage of user-user social relation and user-item ratings. However, user/item diversities in social recommendations are not well utilized in the literature. Especially, inter-factor (social and rating factors) relations and distinct rating values need taking into more consideration. In this paper, we propose an attentive social recommendation system (ASR) to address this issue from two aspects. First, in ASR, Rec-conv graph network layers are proposed to extract the social factor, user-rating and item-rated factors and then automatically assign contribution weights to aggregate these factors into the user/item embedding vectors. Second, a disentangling strategy is applied for diverse rating values. Extensive experiments on benchmarks demonstrate the effectiveness and advantages of our ASR.

1 Introduction

Recommendation system aims to predict unobserved ratings based on users' historical purchases. Users are also involved in social relations where they often acquire and propagate preferences. *Social recommendation* is to leverage the social factor in recommendation systems. It has been verified to be effective for alleviating the data sparsity and cold-start issue existing in traditional collaborative filtering-based recommendation methods (Wu et al. 2019c; Fan et al. 2019b).

In social recommendation, both user-user social relations and user-item rating data are often represented by graph structures. As shown in Figure 1, we structure the user-user relationship as a graph (blue part) and user-item ratings as a bipartite graph (red part) with edge weights representing rating values. Usually, rating values range from "1" to "5" with "1" as dislike and "5" as like most, for instance, in the two benchmark datasets in the experiments Ciao and Epinions. Taking advantage of graph neural network (GNN), e.g., graph convolutional network (GCN) (Kipf and Welling 2017) and graph attention network (GAT) (Veličković et al. 2018), recent works are proposed to extract features from the social graph and the rating graph (Wu et al. 2019c; Fan et al. 2019c,b).

As demonstrated in Figure 1, three factors are often taken into account in social recommendation. *Social factor* reflects that a user's ratings may be influenced by the neighbors in
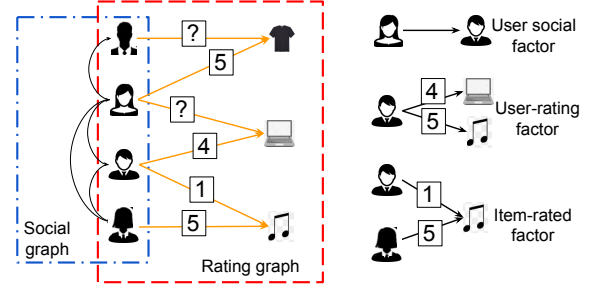

Figure 1: Predict the unobserved ratings in social recommendation by considering user social factor, user-rating factor, and item-rated factor.

Figure 2: Motivation of our attentive social recommendation: considering inter-factor contributions for user/item diversity. ↑ / ↓ means more/less attention weights.

the social graph. In the rating graph, we define *user-rating factor* as the effect on an individual user from all her/his ratings and *item-rated factor* as the impact on an item from all its ratings.

Despite the successes of feature extractors, diversity of users and items are not well investigated in the literature. There still are two main challenges.

First, existing works overlook the different contributions of the multiple factors when considering user/item diversities. Shown in Figure 2, we take four cases for illustration. For users, some users (the first two rows) are easily influenced by friends, i.e., more social factor should be paid attention; while some (the last two rows) have their unique

preferences reflected by their own ratings (more user-rating factor and less social factor). Moreover, evaluating an item could be more subjective or objective. We should recommend a song (subjective items) based on users' different tastes which means that more user-rating factor needs considering. But recommending a computer (objective items) would require more its overall ratings, i.e., more item-rated factor. User/item diversities in real scenarios are more complex than the four exemplars. Thus attentive inter-factor contributions should be emphasized. However, existing methods only separately extract some factor features. For example, *GraphRec* (Fan et al. 2019b) separately applies a GAT in either the social graph or the rating graph and then simply concatenates the extracted features for further rating prediction.

Second, distinct rating values are not well exploited. "like" and "dislike" ratings may propagate in social and rating graphs in different patterns. But existing work, for example, *DANSER* (Wu et al. 2019c) does not distinguish the edge weights (i.e., rating values) in the rating graph and only utilizes rating values in the loss computation.

To tackle the two challenges, in this paper, we propose an *Attentive Social Recommendation (ASR)* model to attentively fuse multiple factors for user/item diversities in social recommendation.

For the first challenge, in ASR, a new graph neural network architecture, Rec-conv layer, is proposed. In each Rec-conv layer, GNNs are applied to extract the three aforementioned factors from the social graph and the rating graph. Attention mechanisms are utilized as well in each Rec-conv layer to automatically assign contribution weights on the three factors and to obtain factor-fused user/item embedding vectors.

For the second challenge, ASR adopts a disentangling strategy to distinguish the propagation of "like" and "dislike" ratings. Specifically, for each rating value, we first induce a subgraph from the entire rating graph. Then we combine all the GNN-extracted user-rating and item-rated factors from each subgraph into the user/item embedding in each Rec-conv layer.

Extensive experimental results on two real-world datasets verify the better effectiveness and efficiency of ASR than state-of-the-art methods. We also conduct ablation study to demonstrate the effectiveness of the inter-factor attention mechanism, the disentangling strategy and GNNs, and to examine the sensitivity of the stacked the Rec-conv layers to the over-smoothing issue (Li, Han, and Wu 2018).

Our major contributions are summarized as follows.

- Diversities of users and items are investigated. Inter-factor contributions and distinct rating propagation are vital to social recommendation.

- A novel attentive social recommendation (ASR) system with stacked Rec-conv layers is proposed to effectively fuse multiple factors for user/item representation learning.

- Extensive experiments on two benchmarks demonstrate the advantages of ASR both in effectiveness and efficiency.

2 Related Work

Classic recommendation system. Collaborating filtering based methods are widely used in recommendation systems (Koren 2010; Koren, Bell, and Volinsky 2009; He et al. 2017). Most methods model a user's preference by collecting and analyzing rating information from other users with matrix factorization techniques. Recently, deep neural networks have been applied in this task (He et al. 2017; Guo et al. 2017; Ying et al. 2018; Wu et al. 2019b; Krishnan et al. 2019; Fan et al. 2019c,a; Jin et al. 2020; Lei et al. 2020). An overview can be found in Zhang et al. (2019).

Social recommendation. Investigating user social relationship in recommendation has drawn increasing attention because of its capability to alleviate the data sparsity and cold-start problem (Tang, Hu, and Liu 2013; Ma et al. 2011; Fan et al. 2019b). For instance, *SocialReg* (Ma et al. 2011) is a matrix factorization method with social regularization. *RSTE* (Ma, King, and Lyu 2009) fuses the users' tastes and their friends influences together with a probabilistic framework. TrustMF (Yang et al. 2016) tries to capture users' reciprocal influence to learn low-dimensional user embedding in truster space and turstee space. SoDimRec (Tang et al. 2016) investigates the heterogeneous relations and weak dependency connections in social graphs. DiffNet (Wu et al. 2019a) introduces an influence propagation mechanism to stimulate the recursive social diffusion process in social recommendation. Attention mechanisms are introduced in DiffNet++ (Wu et al. 2020), EATNN (Chen et al. 2019), DGRec (Song et al. 2019) and SoRecGAT (Vijaikumar, Shevade, and Murty 2019). A survey of social recommendation can be found in Tang, Hu, and Liu (2013).

GNN-based social recommendation. More recently, GNN (Kipf and Welling 2017; Veličković et al. 2018) has been used in social recommendation due to its abilities of aggregating local neighbors information in graphs (Monti, Bronstein, and Bresson 2017; Ying et al. 2018; Wu et al. 2019c; Fan et al. 2019b; Yu et al. 2020). In Monti, Bronstein, and Bresson (2017), the authors generalize GNN to multiple graphs and to learn user/item representations. GraphRec (Fan et al. 2019b) and DSCF (Fan et al. 2019c) apply GATs (Veličković et al. 2018) in the user-user social graph and user-item rating graph separately to extract user/item features. DANSER (Wu et al. 2019c) adopts GAT to learn user/item static and dynamic embedding vectors.

Nevertheless, they cannot effectively and attentively fuse social and rating factors and distinct ratings for user/item diversities in social recommendations.

3 Attentive Social Recommendation

In this section, we introduce the framework of Attentive Social Recommendation which can dynamically extract and fuse social, user-rating and item-rated factors via stacking the newly proposed Rec-conv layers.

3.1 Preliminaries

In this paper, we represent user/item set as $\mathcal{U}$ ($N = |\mathcal{U}|$) and $\mathcal{I}$ ($M = |\mathcal{I}|$), respectively. A social graph $G_S = (\mathcal{U}, \mathcal{S})$ and a user-item rating bipartite graph $G_R = (\mathcal{U}, \mathcal{I}, \mathcal{R})$ are

Table 1: Main symbols

Symbol	Definition
$\mathcal{U}$	the user set; $N = |\mathcal{U}|$
$\mathcal{I}$	the item set; $M = |\mathcal{I}|$
$\mathcal{R}$	the set of observed user-item rating pairs
G_S	the social graph $G_S = (\mathcal{U}, \mathcal{S})$ with $\mathcal{S}$ as the edge set
G_R	the user-item rating bipartite graph $G_R = (\mathcal{U}, \mathcal{I}, \mathcal{R})$
$\mathbf{S}$	the binary adjacency matrix of social graph G_S
$\mathbf{R}$	the integer-rating matrix of G_R ($r_{xy} \in \{1 \cdots K\}$)
$\mathbf{U}^{(l)}$	the user repre. matrix of the l-th Rec-conv layer
$\mathbf{U}^{(l,\mathcal{U} \to u)}$	the social factor repre. matrix of the l-th layer
$\mathbf{U}^{(l,\mathcal{I} \to u)}$	the user-rating factor repre. matrix of the l-th layer
$\mathbf{I}^{(l)}$	the item repre. matrix of the l-th layer
$\mathbf{I}^{(l,\mathcal{U} \to i)}$	the item-rated factor repre. matrix of the l-th layer

given. In G_S, $\mathcal{S}$ describes users social connections. Let its adjacency matrix be $\mathbf{S} \in \{0,1\}^{N \times N}$ with 1/0 representing if social relation exists between two users or not. In G_R, $\mathcal{R}$ records the observed user-item rating pairs. Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ record user-item rating values. A user-item pair $(x, y) \in \mathcal{R}$ ($x \in \mathcal{U}, y \in \mathcal{I}$) is assigned with an existing integer rating value $r_{xy} \in \mathbf{R}$ which ranges from 1 ("like least") to K ("like most"). We also initialize the unobserved ratings in $\mathbf{R}$ with 0s. The goal of social recommendation is to estimate the unobserved ratings. Main notations of this paper are also summarized in Table 1.

3.2 The Framework

ASR targets to attentively fuse three factors (user-user social factor, user-rating factor, and item-rated factor) into user and item embedding. The overall architecture of ASR is shown in Figure 3(a).

We represent users and items with D-dimension latent vectors as $\mathbf{U} \in \mathbb{R}^{N \times D}$ and $\mathbf{I} \in \mathbb{R}^{M \times D}$, respectively. User and item embedding vectors are updated by forwarding through stacked Rec-conv neural network layers.

In each Rec-conv layer, the social factor features are extracted from the social graph G_S; user-rating factor and item-rated factor features are obtained from the bipartite rating graph G_R. In the meanwhile, the user and item embedding vectors are updated by attentively fusing all the three factors.

After obtaining the final user/item embedding, we follow the common setting in the literature (Wu et al. 2019c; Fan et al. 2019b) to predict the unobserved rating value for a user x on an item y. We concatenate the user and item vectors and feed them into an MLP to predict the rating value r'_{xy}. During training, given the ground-truth rating r_{xy}, we take the mean square error as the training objective:

$$Loss = \frac{1}{2|\mathcal{R}'|} \sum_{(x,y) \in \mathcal{R}'} (r'_{xy} - r_{xy})^2 \qquad (1)$$

where $\mathcal{R}' \subset \mathcal{R}$ is the training set containing observed ratings.

4 The Rec-conv Layer

In this section, we introduce our Rec-conv layer to attentively aggregate multiple factors from social and rating graphs in the user/item embedding. A diagram is shown in Figure 3(b). Two main processes in the Rec-conv layer are to update user embedding (Section 4.1) and item embedding Section 4.2.

4.1 Update User Embedding $\mathbf{U}^{(l)} \to \mathbf{U}^{(l+1)}$

Users are influenced by various factors, such as their own tastes (user-rating factor) and social effects from friends (social factor). To model these factors, in the lth Rec-conv layer, we use GNNs to generate two kinds of user latent vectors: *user social vectors* $\mathbf{U}^{(l,\mathcal{U} \to u)}$ and *user-rating vectors* $\mathbf{U}^{(l,\mathcal{I} \to u)}$. An attention mechanism is used to attentively aggregate all factors. In the following, we take GCN (Kipf and Welling 2017) as an example. Note that Other GNNs can also be used, we evaluate their performance in Section 5.4.2.

4.1.1 User Social Vector $\mathbf{U}^{(l,\mathcal{U} \to u)}$
Social vectors are extracted by applying a GCN to propagate local neighbor's information in the social graph. Formally:

$$\mathbf{U}^{(l,\mathcal{U} \to u)} = \sigma(\tilde{\mathbf{S}} \mathbf{U}^{(l)} \mathbf{W}_S^{(l)}) \qquad (2)$$

Here, $\mathbf{U}^{(l)} \in \mathbb{R}^{N \times D}$ is the user latent matrix of the lth layer. $\mathbf{U}^{(l,\mathcal{U} \to u)}$ is the user social factor we aim to extract. $\tilde{\mathbf{S}} = \mathbf{D}_S^{-\frac{1}{2}} (\mathbf{S} + \mathbf{I}_N) \mathbf{D}_S^{-\frac{1}{2}}$ is the modified transition matrix of G_S by adding self-connections ($\mathbf{I}_N$ is the identity matrix) and being normalized by the node-degree diagonal matrix $\mathbf{D}_S$, which is a standard formalization in GCN (Kipf and Welling 2017). In such way, $\tilde{\mathbf{S}} \mathbf{U}^{(l)}$ can aggregate user neighbors influences into the target social factor.

$\mathbf{W}_S^{(l)} \in \mathbb{R}^{D \times D}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, e.g., ReLU. And $\mathbf{U}^{(0)}$ can be randomly initialized or with user profile matrix if available.

4.1.2 User-Rating Vector $\mathbf{U}^{(l,\mathcal{I} \to u)}$
The user-rating vector for a user reflects the user's preference. And it is extracted from all the ratings in the bipartite rating graph. However, a GCN on the entire rating graph G_R will mix distinct rating values which may represent opposite attitudes (e.g., value 1 and 5). So we apply a disentangling strategy to extract diverse rating effects. Note that we also verify the effectiveness of this disentangling strategy in Section 5.4.2.

Specifically, we first induce rating subgraphs based on the K diverse rating values. We let $\mathcal{R}_k = \{(x \in \mathcal{U}, y \in \mathcal{I}) | r_{xy} = k\}$ be the subset of rating-pair with rating $k \in \{1 \cdots K\}$ and $\mathbf{R}_k$ be the corresponding rating matrix. Thus, $\mathcal{R} = \cup_{k=1}^{K} \mathcal{R}_k$, and $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ for $i \neq j$. Then K channels of GCN filters are used. For the kth channel, the extracted user-rating vector is:

$$\mathbf{U}^{(l,\mathcal{I} \to u,k)} = \sigma(\tilde{\mathbf{R}}_k \mathbf{I}^{(l)} \mathbf{W}_R^{(l,k)}) \qquad (3)$$

Here, $\tilde{\mathbf{R}}_k$ is the column-normalized $\mathbf{R}_k$, and $\mathbf{I}^{(l)}$ is the item embedding of the lth Rec-conv layer. Then each row of $\tilde{\mathbf{R}}_k \mathbf{I}^{(l)}$ represents the influence to one user u from all items
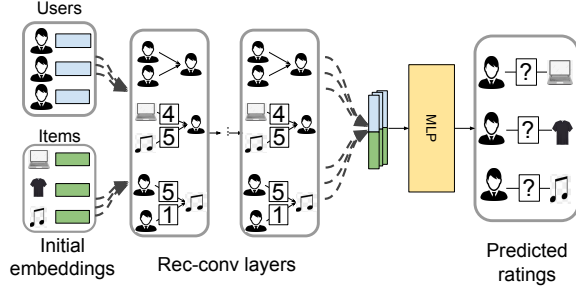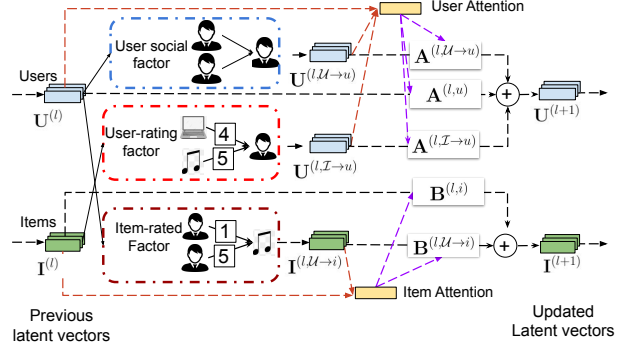
Figure 3: (a) The framework of ASR. Blue/green bars denote user/item latent vectors. (b) Social factor $\mathbf{U}^{(l,\mathcal{U}\to u)}$, user-rating factor $\mathbf{U}^{(l,\mathcal{I}\to u)}$ and item-rated factor $\mathbf{I}^{(l,\mathcal{U}\to i)}$ are extracted and attentively fused in the user/item embedding updates.

rated by u with the rating value k. $\mathbf{W}_R^{(l,k)} \in \mathbb{R}^{D\times D}$ is trainable.

K channels of outputs encode different users rating attitudes. Finally, we concatenate them and project to D-dim. to obtain final user-rating vectors:

$$\mathbf{U}^{(l,\mathcal{I}\to u)} = \sigma(\|_{k=1}^K \mathbf{U}^{(l,\mathcal{I}\to u,k)}\mathbf{W}_{ur}^{(l)}) \qquad (4)$$

where $\|$ is to concatenate matrices horizontally, and $\mathbf{W}_{ur}^{(l)} \in \mathbb{R}^{KD\times D}$ is a projector matrix.

4.1.3 Attentive Aggregation For User Embedding

There are three components to consider (i) $\mathbf{U}^{(l)}$ encodes user own factors from previous layer; (ii) social factor $\mathbf{U}^{(l,\mathcal{U}\to u)}$; and (iii) user-rating factor $\mathbf{U}^{(l,\mathcal{I}\to u)}$.

To differentiate impacts of these factors, we introduce attention mechanisms to control the information aggregated into the updated user embedding $\mathbf{U}^{(l+1)}$ which is the input for the next Rec-conv layer. We let $\mathbf{A}^{(l,u)}$ gate how much previous user embedding information to remember; $\mathbf{A}^{(l,\mathcal{U}\to u)}$ and $\mathbf{A}^{(l,\mathcal{I}\to u)}$ determine importance of the social factor and the user-rating factor.

Without loss of generality, we take $\mathbf{A}^{(l,u)}$ as an example. Calculation of $\mathbf{A}^{(l,\mathcal{U}\to u)}$ and $\mathbf{A}^{(l,\mathcal{I}\to u)}$ are similar. Formally, a single network layer (with parameter $\mathbf{W}_A^{(l,u)}$) is conducted to generate the contribution matrix:

$$\mathbf{A}^{(l,u)} = s([\mathbf{U}^{(l)}\|\mathbf{U}^{(l,\mathcal{U}\to u)}\|\mathbf{U}^{(l,\mathcal{I}\to u)}]\mathbf{W}_A^{(l,u)}) \quad (5)$$

where $s(x) = \frac{1}{1+e^{-x}}$ is the element-wise sigmoid operator.

Then the updated user embedding $\mathbf{U}^{(l+1)}$ (i.e., inputs for the next Rec-conv layer) is obtained as the attention-weighted sum :

$$\begin{aligned}\mathbf{U}^{(l+1)} =&\mathbf{U}^{(l)} \odot \mathbf{A}^{(l,u)} + \mathbf{U}^{(l,\mathcal{U}\to u)} \odot \mathbf{A}^{(l,\mathcal{U}\to u)} \\ &+ \mathbf{U}^{(l,\mathcal{I}\to u)} \odot \mathbf{A}^{(l,\mathcal{I}\to u)}\end{aligned} \qquad (6)$$

where $\odot$ represents element-wise product.

4.2 Update Item Embedding $\mathbf{I}^{(l)} \to \mathbf{I}^{(l+1)}$

To generate a latent vector for an item, not only its own previous item embedding but also the item-rated factor should be considered. We first generate item-rated vector, which encodes information from an item's overall rating. Then the attention mechanism is introduced.

4.2.1 Item-Rated Vector $\mathbf{I}^{(l,\mathcal{U}\to i)}$

The item-rated vector aggregates historical ratings of an item. Similar to the user-rating factor in Section 4.1.2, we adopt K channels of GCN filters for diverse ratings. The output of the kth channel is

$$\mathbf{I}^{(l,\mathcal{U}\to i,k)} = \sigma(\tilde{\mathbf{R}}_k^\intercal \mathbf{U}^{(l)}\mathbf{W}_{RT}^{(l,k)}) \qquad (7)$$

Here, $\tilde{\mathbf{R}}_k^\intercal$ is the column-normalized matrix of the transpose of $\mathbf{R}_k$. Then each row of $\tilde{\mathbf{R}}_k^\intercal \mathbf{U}^{(l)}$ collects effects on one item i from all users who rate i with the rating value k. $\mathbf{W}_{RT}^{(l,k)} \in \mathbb{R}^{D\times D}$ is the trainable transform matrix.

Then we concatenate the K outputs and connect a projection $\mathbf{W}_{ir}^{(l)} \in \mathbb{R}^{KD\times D}$ to get the item-rated vectors:

$$\mathbf{I}^{(l,\mathcal{U}\to i)} = \sigma(\|_{k=1}^K \mathbf{I}^{(l,\mathcal{U}\to i,k)}\mathbf{W}_{ir}^{(l)}) \qquad (8)$$

4.2.2 Attention Aggregation For Item Embedding

To update the item embedding vectors $\mathbf{I}^{(l+1)}$, we introduce item attention mechanisms to assign contribution weights to the previous item vector $\mathbf{I}^{(l)}$ and the item-rated vector $\mathbf{I}^{(l,\mathcal{U}\to i)}$. Similar to the attention in user embedding updating, two item contribution matrices are:

$$\mathbf{B}^{(l,i)} = s([\mathbf{I}^{(l)}\|\mathbf{I}^{(l,\mathcal{U}\to i)}]\mathbf{W}_B^{(l,i)}) \qquad (9)$$

$$\mathbf{B}^{(l,\mathcal{U}\to i)} = s([\mathbf{I}^{(l)}\|\mathbf{I}^{(l,\mathcal{U}\to i)}]\mathbf{W}_B^{(l,\mathcal{U}\to i)}) \qquad (10)$$

where, $\mathbf{W}_B^{(l,i)}, \mathbf{W}_B^{(l,\mathcal{U}\to i)} \in \mathbb{R}^{2D\times D}$ are trainable parameter matrices.

Then we obtain the updated item embedding $\mathbf{I}^{(l+1)}$ by:

$$\mathbf{I}^{(l+1)} = \mathbf{I}^{(l)} \odot \mathbf{B}^{(l,i)} + \mathbf{I}^{(l,\mathcal{U}\to i)} \odot \mathbf{B}^{(l,\mathcal{U}\to i)} \qquad (11)$$

Table 2: Statistics of datasets

Dataset	Ciao	Epinions
# Users ($|\mathcal{U}|$)	7,317	37,311
# Items ($|\mathcal{I}|$)	104,975	36,047
# Ratings ($|\mathcal{R}|$)	111,781	1,054,202
# Social connections ($|\mathcal{S}|$)	283,319	428,034

In summary, in each Rec-conv layer, using attention mechanisms, we dynamically aggregate social factor, user-rating factor and item-rated factor into the user/item embedding vector.

We also emphasize that the attention mechanism help to alleviates the over-smoothing issue in GNN (Li, Han, and Wu 2018). Thus, stacked Rec-conv layers can effectively capture user/item diverse information. Related evaluations are provided in Section 5.4.3.

5 Experimental Study

In this section, we perform extensive experiments to evaluate the performance of the proposed ASR method on two benchmarks, Ciao and Epinions. Social recommendation performances are shown in Section 5.2. Section 5.3 further checks the diversity of predicted ratings. Model evaluations and ablation studies for ASR are followed in Section 5.4. An efficiency evaluation is also provided in Section 5.5.

5.1 Experimental Settings

5.1.1 Datasets And Baselines Two benchmark datasets Ciao and Epinions[1] are used. In these datasets, users can rate and give comments on items. The rating values are integers from 1 (like least) to 5 (like most). Besides, they can also select other users as their trusters. We use the trust graphs as social graphs. The statistics are summarized in Table 2.

We compare ASR with state-of-the-art baselines with publicly available codes, including a traditional recommendation method (*UserMean*, *ItemMean*, and *SVD++*), social recommendation methods (*SocialReg* and *RSTE*), and deep learning-based models (*GraphRec*, *DANSER*). *UserMean* and *ItemMean* predict an unobserved rating with the average of the user's and item's rating values, respectively. *SVD++* (Koren 2010) is a collaborative filtering method considering both explicit and implicit feedback. *SocialReg* (Ma et al. 2011) treats the social graph as regularization and use a matrix factorization framework. *RSTE* (Ma, King, and Lyu 2009) models users' favors and their friends' tastes with a probabilistic factor framework. *GraphRec* (Fan et al. 2019b) uses attention mechanisms to learn user/item embedding separately from social graph and user-item graph. *DANSER* (Wu et al. 2019c) captures user/item dynamic/static features by GATs.

Note that we do not involve and compare with methods for top-N recommendation, such as DiffNet (Wu et al. 2019a, 2020). Because top-N recommendation targets to retrieve or rank N items for users even utilizing temporal information instead of directly predicting user-item rating values. And

[1]https://www.cse.msu.edu/ tangjili/datasetcode/truststudy.htm

the latter is our main purpose of social recommendation in this paper.

5.1.2 Parameter Settings For each dataset, we randomly split 80% existing user-item ratings as the training set, 10% as the validation set for tuning hyperparameters, and the left 10% for testing. For ASR, we set the embedding dimension to 16, the batch size to 4096 and the learning rate to 0.0003. We stack two Rec-conv layers (with GCN architecture[2] and ReLU activation function) followed by an MLP for prediction. Early stopping is also used. For other methods, we follow instructions in their papers to carefully tune hyperparameters, including but not limited to embedding size, batch size and learning rate, and report their best results.

5.2 Performance Comparison

We adopt two widely used metrics, mean absolute error (MAE) and root mean square error (RMSE) (Wu et al. 2019c). Smaller MAE and RMSE scores indicate better performance. We repeat each experiment 5 times and report average results on testing set in Table 3 (first two columns of each dataset). The best performer is highlighted with bold fonts. Note that small improvements in MAE or RMSE will lead to significant enhancement on the performance of the top-N recommendation (Fan et al. 2019b). From the table, we observe:

- Among the matrix factorization-based methods (*SVD++*, *SocialReg*, and *RSTE*), *SVD++* only utilizes user-item rating information, while the two better performers, *SocialReg* and *RSTE*, use the social graph as additional information. Comparing them verifies that social factor can provide complementary information for recommendations.

  Interestingly, we observe that in some cases, the simple method *ItemMean* outperforms *SVD++* in Ciao and Epinions (RMSE). This observation is similar as that pointed out by Dacrema, Cremonesi, and Jannach (2019). They also found that simple methods may achieve comparable performances with more complicated alternatives.

- In general, GNN-based methods, including *ASR*, *GraphRec*, and *DANSER*, outperform traditional social recommendation baselines. Because GNN models can more effectively aggregate information from both social and rating graphs. The comparison between these two types of methods reflects the power of GNN for social recommendation systems.

- *ASR* outperforms others in both datasets with statistical significance. This is because *ASR* considers the diversity of users and items and combine multiple diverse factors. In *ASR*, attention mechanisms can actively extract and aggregate the social, user-rating and item-rated factors from the social and rating graph. The disentangling strategy also differentiates impacts of different rating values. Detailed evaluations are shown in Section 5.4.

[2]We evaluate effectiveness of different GNNs in extracting factors from the social graph and the rating graph in Section 5.4.2.

Table 3: Performance results on testing set

Methods	Ciao			Epinions		
	MAE↓	RMSE↓	P-ACC↑	MAE↓	RMSE↓	P-ACC↑
UserMean	0.783	1.033	–	0.951	1.220	–
ItemMean	0.749	1.016	–	0.831	1.081	–
SVD++	0.768	1.043	–	0.821	1.096	–
RSTE	0.753	0.999	0.548	0.828	1.077	0.714
SocialReg	0.746*	0.986*	0.558*	0.825*	1.064*	0.717*
DANSER	0.740*	0.990*	0.630*	0.814*	1.076*	0.722*
GraphRec	0.745*	0.983*	0.601*	0.820*	1.075*	0.719*
ASR	**0.717**	**0.962**	**0.637**	**0.781**	**1.048**	**0.730**

↓/↑ next to metrics means that the smaller/larger, the better.
* We do student-t test between ASR and the best baselines on each metric (* corresponds to $p < 0.05$).
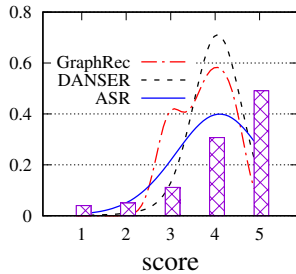

Figure 4: Distributions of predicted ratings in Ciao test set

(a) MAE   (b) RMSE

Figure 5: Impact of user/item attention mechanism

5.3 Diversity of User-Item Ratings

Except for ASR's better results on MAE and RMSE, in this section, we demonstrate that ASR can obtain more accurate and diverse ratings than baselines.

Because MAE and RMSE are both for the overall performance, we now check whether a model can predict accurate ratings for each user with the *pairwise-ranking accuracy (P-ACC)*. Suppose that one user has distinct ratings for two items, we define *a hit* when one algorithm can predict the ratings with the correct relative ranking. Then P-ACC reflects the overall hitting rate. Higher P-ACC means better performance. Table 3 (the third column of each dataset) shows the P-ACC results over the testing set. Note that we only consider the top-5 performers. Results evidence that ASR can predict more accurate individual rating-rank.

Next, we check the goodness of overall rating distributions of ASR and the other two best baselines *DANSER* and *GraphRec*. The bars in Figure 4 are the ground-truth rating histogram of Ciao testing set. We can see that *GraphRec* predicts most ratings as "3" and "4" and neglects other values even for the most value "5" in the ground-truth. *DANSER* even narrows its all ratings around "4" (values in [3.6, 4.3] covers more than 80% of its ratings). However, *ASR* can fit better with the ground-truth by considering the rating diversity.

5.4 Model Analysis

In this section, we conduct ablation study for *ASR*. Three aspects are evaluated: the necessity of the two main components of *ASR*, i.e., inter-factor attention mechanisms and the
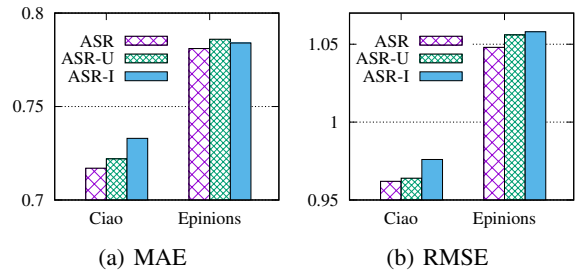
disentangling strategy; and the sensitivity of *ASR* to over-smoothing of GNNs (Li, Han, and Wu 2018).

5.4.1 Impact of Inter-Factor Attention Mechanisms
To capture inter-factor contribution, we apply attentions in user/item embedding updating. To show the effectiveness of the attention, we compare *ASR* with two variants:

- *ASR-U* removes the user attention, i.e., $\mathbf{A}^{(l,u)}$, $\mathbf{A}^{(l,\mathcal{U}\to u)}$ and $\mathbf{A}^{(l,\mathcal{I}\to u)}$ in Equation (6) are set with all 1s.

- *ASR-I* removes the item attention. i.e., $\mathbf{B}^{(l,i)}$ and $\mathbf{B}^{(l,\mathcal{U}\to i)}$ in Equation (11) are set with all 1s.

MAE and RMSE results of ASR and its two variants are shown in Figure 5. We can see that *ASR* achieves the best MAE and RMSE scores in both Ciao and Epinions. Comparing *ASR* with *ASR-U* and *ASR-I*, we conclude that including user/item inter-factor attentions provides *ASR* powerful and flexible abilities to effectively aggregate impacts from multiple factors, which leads to better results.

5.4.2 Impact of The Disentangling Strategy And GNNs
The disentangling strategy of *ASR* first splits the entire rating graph into subgraphs based on distinct rating values, and then aggregates GNN-extracted features from each subgraph. To verify its advantage, we compare *ASR* with its variant: *ASR-w/-G_R-GAT* which ignores diverse rating values and adopts GAT on the entire rating graph G_R. In *ASR-w/-G_R-GAT*, though rating value differences are removed in G_R, GAT can assign distinct weights on different items for one user or different users for one item. Note that GAT

Table 4: ASR v.s. its variants

Methods	Ciao		Epinions	
	MAE↓	RMSE↓	MAE↓	RMSE↓
ASR	0.717	0.962	0.781	1.048
ASR-w/-G_R-GAT	0.789	1.027	0.803	1.076
ASR-w/-G_S-GAT	0.720	0.971	0.793	1.058

(a) MAE   (b) RMSE   (c) Distance
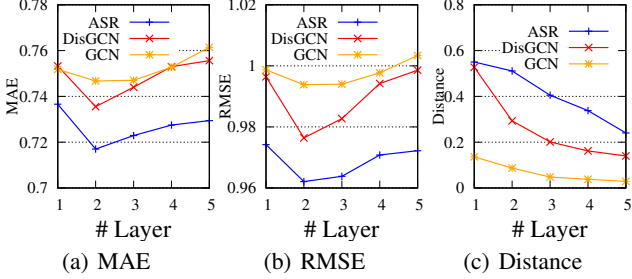
Figure 6: Impacts of stacking Rec-conv layers on Ciao dataset

Table 5: Training time (sec) per epoch

Method	Ciao	Epinion
DANSER	76.372	272.78
GraphRec	862.10	1786.37
ASR	15.53	28.90

is also used in both the two most competitive baselines *DANSER* and *GraphRec*.

Comparing the first two rows in Table 4, *ASR* outperforms *ASR-w/-G_R-GAT* significantly. Although the GAT in *ASR-w/-G_R-GAT* aggregates neighbors' features with different weights in the rating graph, it cannot effectively leverage the diverse rating scores. However, the disentangling strategy in *ASR* guarantees that diverse propagating patterns of diverse ratings can be distinguished.

To check the effect of different GNN, in *ASR*, we also replace the GCN which is applied in the social graph G_S with a GAT but keep the disentangling strategy in the rating graph G_R (*ASR-w/-G_S-GAT*). Comparing *ASR* with *ASR-w/-G_S-GAT* in Table 4, we see that applying GAT in the social graph G_S can obtain comparable performances as the original ASR.

Comparing the three together, we conclude that the disentangling strategy in the rating graph plays an essential role in ASR. Both GAT and GCN can be applied in the social graph to effectively extract the user-social factor.

5.4.3 Stacking Rec-conv Layers V.S. Over-smoothing
With more GNN layers stacked, most GNNs suffer from over-smoothing issue (Li, Han, and Wu 2018), which means that embedding vectors tend to be similar. If we compute the pairwise-distance (e.g., cosine distance) of the embedding vectors, then the more layers are, the smaller the average distance is. The larger the distance is, the less the over-smoothing is. To evaluate the sensitivity of *ASR* to the over-smoothing, we compare *ASR* with its two GNN variants (i) DisGCN: removing the attentions in *ASR*; and (ii) GCN: removing attentions and disentangling in *ASR*. Results over different numbers of stacked layers are shown in Figure 6. The results demonstrates that *ASR* can alleviate the over-smoothing more than the two variants.

ASR can alleviate the over-smoothing for two reasons. First, comparing *ASR* with DisGCN, user/item attentions can actively control how much impacts from different fac-

tors to pass forward to the next layer. Besides, comparing GCN with DisGCN and *ASR*, the disentangling strategy splits the mixed ratings in the rating graph and diversifies effects of different ratings in the user/item embedding updating processes. As a result, we can stack multiple Rec-conv layers when encountering complex datasets to aggregate more underlying factor features.

5.5 Efficiency Evaluation
In *ASR*, GCNs are applied for distinct rating values. This brings in more parameters than when considering the rating graph as a whole graph. However, in practice, we found that *ASR* executes very fast. We compare the training time per epoch of *ASR* and the two deep learning based baselines *DANSER* and *GraphRec*. Note that we use the same embedding dimension for the three models. As shown in Table 5, *ASR* is the most efficient model. We analyze reasons as the following. Multiple GATs are used in both *DANSER* and *GraphRec*. Different with GCN, attention weights computation in GAT costs quadratic time in terms of number of nodes. Besides, a policy network exists in *DANSER* to search optimal returns. In *GraphRec*, the current implementation[3] takes each user-item rating pair as a minibatch, because it specifies individual user who may have different number of user-neighbors and rated items. This means that trainable parameters will be updated for each rating pair which is time-consuming.

6 Conclusion
In this paper, we proposed an attentive social recommendation method *ASR*. *ASR* includes user and item attentions to capture diversities of users and items. The proposed Rec-conv network layer and attention mechanisms enable *ASR* to actively extract and fuse the social factor, user-rating factor and item-rated factor from user social graph and user-item rating graph. In addition, a disentangling strategy was developed to aggregate information from diverse ratings in the user-item rating graph. Comprehensive experiments on two benchmark datasets demonstrate the effectiveness of *ASR*. Ablation studies on attention mechanisms, disentangling strategy and GNNs verified the necessity of these modules. We can stack multiple Rec-conv layers in *ASR* but less being affected by the over-smoothing. The training process of *ASR* is much faster than alternatives as well.

---

[3]https://github.com/wenqifan03/GraphRec-WWW19

References

Chen, C.; Zhang, M.; Wang, C.; Ma, W.; Li, M.; Liu, Y.; and Ma, S. 2019. An efficient adaptive transfer neural network for social-aware recommendation. In *SIGIR*, 225–234.

Dacrema, M. F.; Cremonesi, P.; and Jannach, D. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *RecSys*, 101–109.

Fan, W.; Derr, T.; Ma, Y.; Wang, J.; Tang, J.; and Li, Q. 2019a. Deep adversarial social recommendation. In *IJCAI*, 1351–1357. AAAI Press.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019b. Graph Neural Networks for Social Recommendation. In *WWW*, 417–426. ACM.

Fan, W.; Ma, Y.; Yin, D.; Wang, J.; Tang, J.; and Li, Q. 2019c. Deep social collaborative filtering. In *RecSys*.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*, 1725–1731.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW*, 173–182. International World Wide Web Conferences Steering Committee.

Jin, B.; Gao, C.; He, X.; Jin, D.; and Li, Y. 2020. Multi-behavior Recommendation with Graph Convolutional Networks. In *SIGIR*, 659–668.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Koren, Y. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD* 4(1): 1.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* .

Krishnan, A.; Cheruvu, H.; Tao, C.; and Sundaram, H. 2019. A modular adversarial approach to social recommendation. In *CIKM*, 1753–1762. ACM.

Lei, Y.; Wang, Z.; Li, W.; Pei, H.; and Dai, Q. 2020. Social Attentive Deep Q-networks for Recommender Systems. *TKDE* .

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

Ma, H.; King, I.; and Lyu, M. R. 2009. Learning to recommend with social trust ensemble. In *SIGIR*, 203–210. ACM.

Ma, H.; Zhou, D.; Liu, C.; Lyu, M. R.; and King, I. 2011. Recommender systems with social regularization. In *WSDM*, 287–296. ACM.

Monti, F.; Bronstein, M.; and Bresson, X. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *NeurIPS*, 3697–3707.

Song, W.; Xiao, Z.; Wang, Y.; Charlin, L.; Zhang, M.; and Tang, J. 2019. Session-based social recommendation via dynamic graph attention networks. In *WSDM*, 555–563.

Tang, J.; Hu, X.; and Liu, H. 2013. Social recommendation: a review. *Social Network Analysis and Mining* 3(4): 1113–1133.

Tang, J.; Wang, S.; Hu, X.; Yin, D.; Bi, Y.; Chang, Y.; and Liu, H. 2016. Recommendation with social dimensions. In *AAAI*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Vijaikumar, M.; Shevade, S.; and Murty, M. N. 2019. SoRecGAT: Leveraging Graph Attention Mechanism for Top-N Social Recommendation. In *ECML/PKDD*, 430–446. Springer.

Wu, L.; Li, J.; Sun, P.; Ge, Y.; and Wang, M. 2020. DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation. *arXiv preprint arXiv:2002.00844* .

Wu, L.; Sun, P.; Fu, Y.; Hong, R.; Wang, X.; and Wang, M. 2019a. A neural influence diffusion model for social recommendation. In *SIGIR*, 235–244.

Wu, Q.; Jiang, L.; Gao, X.; Yang, X.; and Chen, G. 2019b. Feature evolution based multi-task learning for collaborative filtering with social trust. In *IJCAI*, 3877–3883. AAAI Press.

Wu, Q.; Zhang, H.; Gao, X.; He, P.; Weng, P.; Gao, H.; and Chen, G. 2019c. Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. In *WWW*, 2091–2102. ACM.

Yang, B.; Lei, Y.; Liu, J.; and Li, W. 2016. Social collaborative filtering by trust. *TPAMI* 39(8): 1633–1647.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, 974–983. ACM.

Yu, J.; Yin, H.; Li, J.; Gao, M.; Huang, Z.; and Cui, L. 2020. Enhance Social Recommendation with Adversarial Graph Convolutional Networks. *arXiv preprint arXiv:2004.02340* .

Zhang, S.; Yao, L.; Sun, A.; and Tay, Y. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52(1): 5.