An Explainable Recommendation Method Based on Multi-timeslice Graph Embedding

Huiying Wang, Yue Kou(✉), Derong Shen, and Tiezheng Nie

Northeastern University, Shenyang 110004, China
307326704@qq.com, {kouyue, shenderong, nietiezheng}@cse.neu.edu.cn

Abstract. Deep neural networks (DNN) can be used to model users' behavior sequences and predict their interest based on the historical behavior. However, current DNN-based recommendation methods lack explainability, making them difficult to guarantee the credibility of the recommendation results. In this paper, a Multi-Timeslice Graph Embedding (MTGE) model is proposed. First, it can effectively obtain the embedded representations of user behavior (or items) on a single timeslice. Second, the dynamic evolution of user preferences can be analyzed through integrating the embedded representations on multi-timeslices. Then, an explainable recommendation algorithm based on MTGE is proposed, which can effectively improve the accuracy of recommendation and support the model-level explainability. The feasibility and effectiveness of the key technologies proposed in the paper are verified through experiments.

Keywords: Explainable recommendation · Multiple timeslices · Graph embedding · User behavior sequences

1 Introduction

Personalized recommendation systems have played an increasingly important role in people's lives. Among a large number of recommendation algorithms, the collaborative filtering algorithms has been widely used, but traditional collaborative filtering algorithms are less accurate. Deep neural networks (DNN) can be used to capture deep features of users and items to get more accurate representations. However, DNN is often seen as a black box with the disadvantages of unexplainable processing.

Let us consider the following motivating scenarios.

Example 1.1: A month ago, a user gave a high rating to a smartwatch, while a week earlier he was interested in a tennis racket. That is, the user's interests are not static. So, we need to describe the dynamic evolution of user preferences.

Example 1.2: Fig. 1 depicts DNN-based recommendation model and explainable recommendation model. DNN-based recommendation model does not support explainability, and it is difficult to convince users. On the contrary, the explainable recommendation model can give supporting arguments for the results. Some reasons such as "N users love this item" that can reflect the characteristics of the item. It can help people understand the model and improve its transparency.

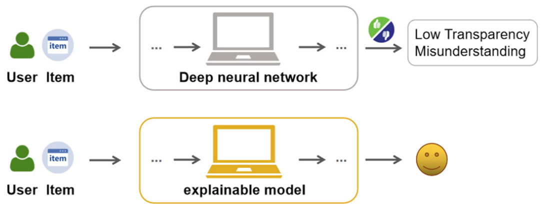


Fig. 1. Comparison of recommendation models

In this paper, we propose an explainable recommendation method based on multi-timeslice graph embedding model. Our main contributions are summarized as follows:

- A Multi-Timeslice Graph Embedding model (namely MTGE) is proposed. First, it can effectively obtain the embedded representations of user behavior (or items) on a single timeslice. Second, the dynamic evolution of user preferences can be analyzed through integrating these embedded representations on multiple timeslices.
- An explainable recommendation algorithm based on MTGE is proposed, which can effectively improve the accuracy of recommendation and support the model-level explainability. On the one hand, the ratings are predicted based on the user's latent vectors and item latent vectors. On the other hand, the explainability of the model is supported, which can effectively improve the transparency of the recommendation model.
- Compared with existing methods through experiments, the effectiveness of our method on real datasets is demonstrated.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 defines several important concepts used in the paper. Section 4 proposes our multi-timeslice graph embedding model. Section 5 proposes an explainable recommendation algorithm based on MTGE. Section 6 shows the experimental results. Section 7 concludes the paper and presents the future work.

2 Related Work

Many methods of recommendation have been proposed in recent years. First, we review the techniques for them. Then we analyze how our work differs from them.

The collaborative filtering recommendation algorithm [1–5, 7] has been widely researched and used in recent years. PMF [2] models the latent vectors of users and items through a Gaussian distribution. Ma Porteous et al. [5] added auxiliary information to Bayesian matrix factorization to improve the accuracy of the recommendation results.

Deep neural networks (DNN) [6, 8] have significant feature learning capabilities by learning deep network structures to represent information about users and items. NeuMF [6] presented a Neural Collaborative Filtering framework to learn latent features of users and items by multi-layer perceptron. Some works have used DNN modeling users' behavior sequences to predict users' preferences [12, 14, 15]. RRN

[12] used a recurrent neural network (RNN) for the temporal recommendation, it can capture the dynamic changes of users. M3 [14] combined the RNN model and attentional to model the long-term sequence of user behaviors.

More recently, deep neural network (DNN) techniques in graph data [13, 18] have made great developments. These deep neural network architectures are known as graph neural networks (GNNs) [9–11, 17], which are used to learn meaningful representations of graph data. Vaswani et al. [16] proposed a graph attention network to resolve the shortcomings of existing methods based on graph convolution. Fan et al. [17] proposed a graph neural network framework for rating predictions that models social graphs to learn user representations.

In previous work, traditional recommendation techniques have mainly considered the static preferences of users. Although some works used DNN to capture users' dynamic preferences, DNN are unexplainable, making the model difficult to understand. In this paper, we proposed the multi-timeslice graph embedding model to obtain the user dynamic preferences and MTGE-based explainable recommendation algorithm can support the model-level explainability.

3 Problem Definition

User-Item Graph. $\mathbf{R} \in \mathbf{R}^{n \times m}$ is user-item graph, also known as the user-item rating matrix, where n is the number of users and m is the number of items, r_{ij} is the rating score of user u_i for item v_j, which can be regarded as the opinion of u_i for v_j.

Latent Vector. The latent vectors of users and items can be regarded as their hidden features, which can predict the user's preference for items.

Temporal Recommendation. Given a user set U, an item set V, where the user-item preference sequences from time 1 to time T is: $\mathbf{R} = \left[\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^T,\right]$. The goal of temporal recommendation is to predict the behavior of each user at time $T + 1$.

Rating Prediction. The rating prediction is based on the user-item rating matrix $\mathbf{R}$, $\hat{r}_{ij}$ represents the predicted rating of u_i for v_j, and we aim to predict the missing rating value in $\mathbf{R}$.

4 MTGE Model

In this section, we describe the model of MTGE. First, we introduce the overall framework of the model, then the single-timeslice graph embedding method is explained, finally, the integration method of multi-timeslice graph embedding is proposed.

4.1 Model Overview

The basic idea of our MTGE model is shown as Fig. 2. The model consists of three components: user embedding, item embedding and rating prediction.

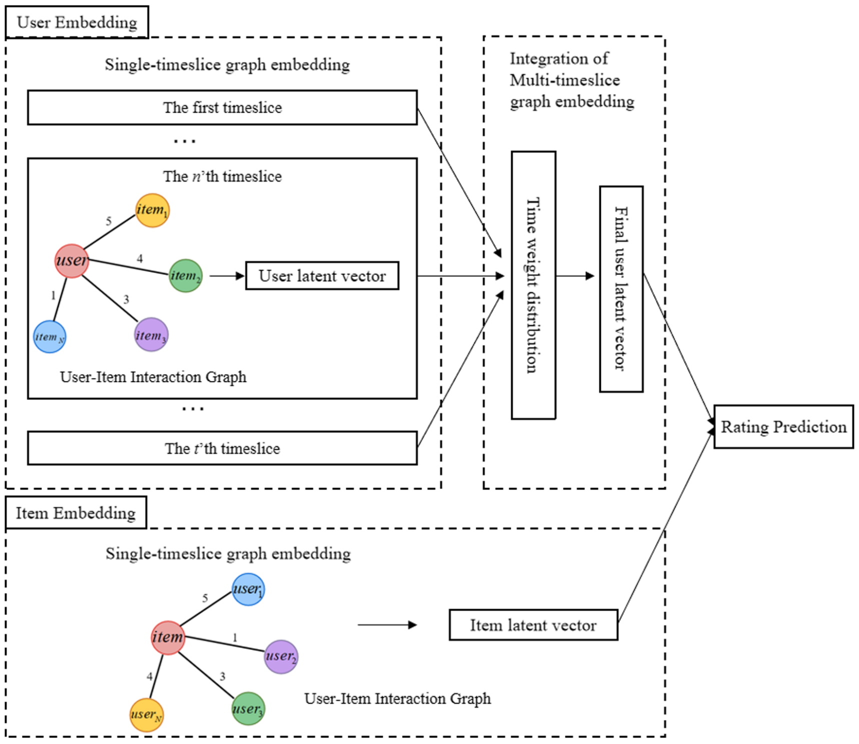


Fig. 2. Multi-timeslice graph embedding model

- User embedding. It is to obtain the user latent vectors. The features of users can be reflected in a set of items that users interact with. Considering the dynamic features of user preferences, the sequence of user behavior is first divided into T timeslices in time sequence.
- Item embedding. It is to obtain the item latent vectors. Similarly, item latent vector can be learned through a set of users that the item has interacted with. And since the item features are not easy to change in a short time, we regard them as static.
- Rating prediction. It is to get the user's predicted rating for the item. We integrate user and item modeling to predict ratings and learn model parameters.

4.2 Single-Timeslice Graph Embedding

The graph neural network updates the current node status by aggregating the information of neighboring nodes. Therefore, fusing neighborhood information from the user-item graph, and the user latent vector is learned through the set $N(i)$ of items that the user has interacted with. The latent vector $\mathbf{h}_i$ of user u_i can be expressed as:

$$\mathbf{h}_i = ReLU\left(\mathbf{W} \cdot \left\{\sum\nolimits_{k \in N(i) \cup \{i\}} \alpha_{ik} \mathbf{o}_{ik}\right\}\right) \tag{1}$$

where *ReLU* is the non-linear activation function and $\mathbf{W}$ is the weight matrix. $\mathbf{o}_{ik}$ is the opinion interaction vector between u_i and v_k, which is obtained by aggregating the embedding vector of v_k and the rating embedding vector. Since the user's rating of items can reflect the user's preference, encoding the interaction information into the latent vector can get a higher accuracy vector. To distinguish the influence of each item on user u_i, α_{ik} denotes the attention weight of the interaction with v_k in contribution to $u_i's$ latent vector from the interaction history $N(i)$. And the attention network is defined as:

$$\alpha_{ik}^{*} = ReLU\left(\mathbf{W}_2 \cdot RELU\left(\mathbf{W}_1\left[\mathbf{e}_u^i; \mathbf{o}_{ik}\right]\right)\right) \tag{2}$$

where the inputs of the attention network is the embedded vector $\mathbf{e}_u^i$ of the target user u_i and opinion interaction vector $\mathbf{o}_{ik}$. Then the final attention scores are obtained by normalizing α_{ik}^{*} using the Softmax function:

$$\alpha_{ik} = \frac{\exp\left(\alpha_{ik}^{*}\right)}{\sum_{k \in N(i)} \exp\left(\alpha_{ik}^{*}\right)} \tag{3}$$

Likewise, we use a similar method to learn the latent vector of the items. For each item, information can be collected from a group of users (denoted as $D(j)$) who interacted with. The latent vector $\mathbf{z}_j$ for item v_j is:

$$\mathbf{z}_j = ReLU\left(\mathbf{W} \cdot \left\{\sum_{k \in D(j) \cup \{j\}} \alpha_{jt}\mathbf{s}_{jt}\right\}\right) \tag{4}$$

where $\mathbf{s}_{tj}$ is the opinion interaction vector of v_j with u_t and α_{jt} represents the attention weight to identify the importance of different users.

$$\alpha_{jt}^{*} = ReLU\left(\mathbf{W}_2 \cdot ReLU\left(\mathbf{W}_1[\mathbf{e}_v^j; \mathbf{s}_{jt}]\right)\right) \tag{5}$$

$$\alpha_{jt} = \frac{exp\left(\alpha_{jt}^{*}\right)}{\sum_{k \in D(j)} exp\left(\alpha_{jt}^{*}\right)} \tag{6}$$

4.3 Integration of Multi-timeslice Graph Embedding

User preferences for items are not always static, but can change over time. Therefore, user's preferences are summarized as a matrix sequence $\mathbf{R} = \left[\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^T,\right]$. For a single timeslice, use the graph embedding method introduced in Sect. 4.2 to obtain the latent factor of user u_i on each timeslice: $\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, \ldots, \mathbf{h}_{i,T}$.

However, even if a user's recent behavior is more likely to influence the current decision, the user's distant behavioral preferences can also influence current behavior to some extent. As shown in Fig. 3, we introduce a time decay function to artificially control the weight of each timeslice' influence on the current user's interest preferences:

$$\beta_t^* = e^{-\alpha t} \tag{7}$$

$$\beta_t = \frac{\exp(\beta_t^*)}{\sum_{t \in T} \exp(\beta_t^*)} \tag{8}$$

where β_t^* is the value of β at moment t and α is the "cooling factor", which takes a smaller value if wants to slow down the rate of decay of interest, otherwise, it takes a larger value. Then, the final time weight is normalized with a Softmax function.

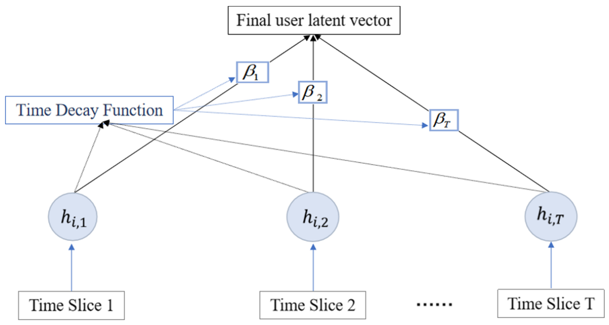


Fig. 3. Time weight distribution

For the weighted sum of the representations of user u_i in each timeslice, the final interest representation $\mathbf{H}_i$ of u_i can be obtained:

$$\mathbf{H}_i = \sum_{t=1}^{T} \beta_t \cdot \mathbf{h}_{i,t} \tag{9}$$

5 MTGE-Based Explainable Recommendation Algorithm

In this section, an explainable recommendation algorithm based on MTGE is proposed. We first introduce the algorithm, then the rating prediction and parameter learning section of the algorithm is represented, and finally analyze the explainability of the algorithm.

5.1 Algorithm Description

Training Stage. First, construct the MTGE model with the training set as input. Then initialize the model parameters, and calculate the predicted ratings of the model. Next, the model parameters are updated by gradient descent to minimize the loss function.

Testing Stage. First, construct the MTGE model with the testing set as input. Then the ratings were calculated using the trained model parameters as the final ratings predicted output.

MTGE-based explainable recommendation algorithm is showed in Algorithm 1, and the major steps are as follows.

Step 1 (line 1). Construct MTGE model based on rating matrix and the number of timeslices.
Step 2 (line 2–4). Divide the rating matrix **R** according to the number of timeslices T, initialize the user embedding vector matrix U, item embedding matrix V, score embedding matrix S and $loss$ value.
Step 3 (line 5–13). Calculate user latent vector $\mathbf{H}_i$ and item latent vector $\mathbf{z}_j$.
Step 4 (line 14–15). Learning Stage. Calculate the rating error, update the model parameters by gradient descent until the stop condition is satisfied, and then stop the iteration.
Step 5 (line 17–19). Generate prediction results. For the users and items to be predicted, after obtaining the model parameters through training, the rating prediction is performed based on the user latent vector and the item latent vector.

Algorithm 1: MTGE-based explainable recommendation algorithm

```
Input:  Rating matrix $\mathbf{R}$, Number of timeslices $T$
Output: Predicted rating $\widehat{\mathbf{R}}$
1.   Construct MTGE based on $\mathbf{R}$, $T$
2.   $\mathbf{R} = [\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^T,]$
3.   Initialize $U, V, S$, the neural network parameter
4.   Initialize $loss$
5.   For $u_i \in U$ do
6.      For each timeslice do
7.      $\mathbf{h}_{i,t} = compu(\mathbf{R}^t, U, V, S)$
8.      End for
9.      $\mathbf{H}_i = \sum_{t=1}^{T} \beta_t \cdot \mathbf{h}_{i,t}$
10.  End for
11.  For $v_j \in V$ do
12.     $\mathbf{z}_j = compu(\mathbf{R}^t, U, V, S)$
13.  End for
14   Compute $loss$
15.  Update $U, V, S$, the neural network parameter
16.  IF meet the stop condition then
17.  $\hat{r}_{ij} = MLP[\mathbf{H}_i; \mathbf{z}_j]$
18.  End IF
19.  Return $\widehat{\mathbf{R}}$
```

5.2 Rating Prediction and Parameter Learning

Rating Prediction. The latent vectors of users and items can reflect their preferences and features. We can first connect $\mathbf{H}_i$ and $\mathbf{z}_j$, then feed it into multilayer perceptron (MLP) to obtain the rating prediction:

$$\hat{r}_{ij} = MLP\left[\mathbf{H}_i; \mathbf{z}_j\right] \tag{10}$$

Parameter Learning. To learn the model parameters for MTGE, we specify a loss function for optimization. The loss function is expressed as:

$$loss = \frac{1}{2|T|}\sum\nolimits_{(u,i)\in T}\left(r_{ui} - \hat{r}_{ij}\right)^2 \tag{11}$$

5.3 Explainability Analysis

This paper consider that user preferences for items are change over time. Some researchers have used deep neural networks (DNN) to model dynamic user preferences, but the model is unexplainable, reducing the transparency of the recommended model.

Since the user's behavior can reflect his or her current preferences, we split the user behavior data based on timeslice and model the user behavior separately to obtain the user preference representation on each timeslice. Next, combining the user's interest decay phenomenon, and then weighing and integrating the user's preference representation in each timeslice, which makes the user's final feature vector have intuitive meaning. For example, a user's behavior a week ago usually has more influence on his current preferences than his behavior six months ago. Moreover, the model-level explainability has multiple significance for the recommendation system, which can improve the transparency and persuasiveness of the system.

6 Experiments

6.1 Dataset

We choose the Epinions and the Ciao dataset, where users can score the products. The statistics of the two datasets are shown in Table 1.

Table 1. Statistics of the experimental datasets

Dataset	Epinions	Ciao
Users	136	103
Items	7716	994
Ratings	12659	3016
Time windows	12	6

6.2 Evaluation Metrics

For the rating prediction task of the recommended model, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are usually used to evaluate the prediction accuracy:

$$RMSE = \sqrt{\frac{1}{|T|}\sum\nolimits_{(u,i)\in T}\left(r_{ui} - \hat{r}_{ui}\right)^2} \tag{12}$$

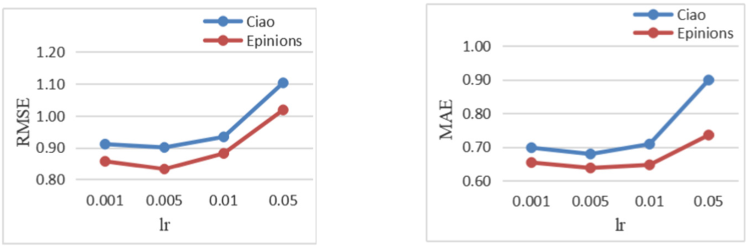
$$MAE = \frac{1}{|T|}\sum\nolimits_{(u,i)\in T}\left(r_{ui} - \hat{r}_{ui}\right) \tag{13}$$

where r is the true rating data from the testing set, $\hat{r}$ is the predicted rating of the recommended system, and T is the testing set.

6.3 Performance Evaluation

Hyperparameter Settings. To obtain the optimal combination of parameters, we experimented with the main parameters.

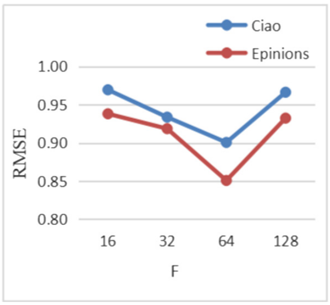
Learning Rate. Figure 4(a) and 5(b) shows the changes in the RMSE and MAE at different learning rates. It can be seen that when the learning rate is 0.005, the RMSE and MAE are lowest on the two datasets. Therefore, we set the learning rate to 0.005.



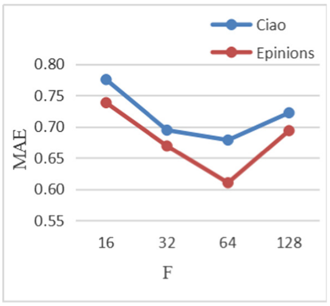
(a) RMSE (b) MAE

Fig. 4. The effect of learning rate of MTGE algorithm

Embedded Vector Dimension. Figure 5(a) and 5(b) shows the changes in the RMSE and MAE at different embedding vector dimensions. It can be seen that when the embedding vector length is 64, the RMSE and MAE are lowest on the two datasets. Therefore, we set the embedding vector length to 64.



(a) RMSE



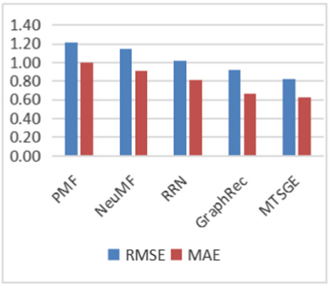
(b) MAE

Fig. 5. The effect of embedding vector length of MTGE algorithm

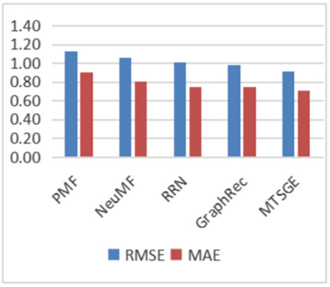
Baseline Algorithm. In order to evaluate the performance of the algorithm, we compare our algorithm with four baseline methods:

- PMF: A probability matrix factorization model that uses a rating matrix to model latent vectors of users and items.
- NeuMF: A matrix factorization model and neural network architecture, using a deeper neural network can provide better recommendation performance.
- RRN: It used a recurrent neural network (RNN) for the temporal recommendation, this model can capture the changes in users' dynamic preferences.
- GraphRec: It proposed a state-of-the-art graph neural network model, which can model graph data coherently for rating prediction.

Figure 6(a) and 6(b) shows the performance of methods. The PMF algorithm is a traditional matrix factorization method, while NeuMF is based on a neural network architecture. RRN takes into account the user's dynamic interests, and its performance has been significantly improved. The GraphRec algorithm good behavior implies the power of graph neural networks in node learning. The MTGE algorithm performs well in both RMSE and MAE, shows that using the graph neural network and considering the user's dynamic features can improve the recommendation performance.



(a) Epinions

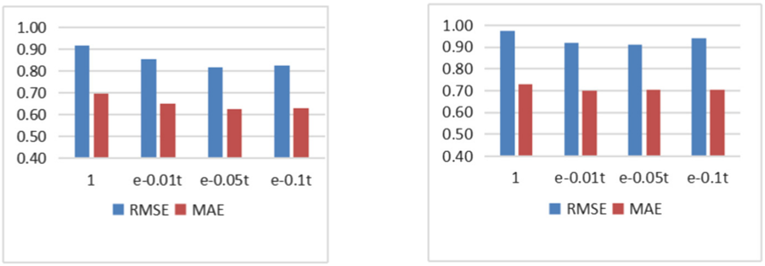


(b) Ciao

Fig. 6. Performance comparison between MTGE algorithm and other algorithms

Explainability Assessment. To better understand the temporal explainability of the MTGE algorithm, the recommended performance under different time functions is compared.

Figure 7(a) and 7(b) shows the results on the different time functions. The exponential function represents that the user's interest decreases exponentially with time. It can be seen that considering the time factor can improve the recommendation performance.



(a) Epinions (b) Ciao

Fig. 7. Performance comparison of different time functions

For example, for the user preference representation within each timeslice, the closer the timeslice is to the current point in time, the greater the contribution of the timeslice to predicting the user's current interest, while the impact weight of the timeslice faraway is relatively reduced, i.e., the user's current interest is more dependent on his recent behaviors.

7 Conclusion

In this paper, a multi-timeslice graph embedding (MTGE) model is proposed, and the model considers the dynamics of users' interest preferences. Besides, this paper proposes an explainable recommendation algorithm based on MTGE, which has better results compared with existing methods. In the future, we will analyze the user interest drift on multiple timeslices, then improve the model based on user psychology.

Acknowledgment. This work is supported by the National Key R&D Program of China (2018YFB1003404) and the National Natural Science Foundation of China (61672142).

References

1. Wu, Y., Ester, M.: FLAME: a probabilistic model combining aspect based opinion mining and collaborative filtering. In: Proceedings of the ACM International Conference on Web Search and Data Mining, WSDM, pp. 199–208 (2015). https://doi.org/10.1145/2684822.2685291

2. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Proceedings of the International Conference on Neural Information Processing Systems, NIPS, pp. 1257–1264 (2007). https://doi.org/10.3233/ifs-141462
3. Ma, H., Yang, H., Lyu, M. R., King, I.: SoRec: social recommendation using probabilistic matrix factorization. In: Proceedings of the ACM Conference on Information and Knowledge Management, CIKM, pp. 931–940 (2008). https://doi.org/10.1145/1458082.1458205
4. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, pp. 426–434 (2008). https://doi.org/10.1145/1401890.1401944
5. Porteous, I., Asuncion, A.U., Welling, M.: Bayesian matrix factorization with side information and Dirichlet process mixtures. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 563–568 (2010). https://doi.org/10.5555/2898607.2898698
6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: Proceedings of the International Conference on World Wide Web, WWW, pp. 173–182 (2017). https://doi.org/10.1145/3038912.3052569
7. Xiong, X., Zhang, M., Zheng, J., Liu, Y.: Social network user recommendation method based on dynamic influence. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 455–466. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_42
8. Kim, D., Park, C., Oh, J., Lee, S., Yu, H.: Convolutional matrix factorization for document context-aware recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 233–240 (2016). https://doi.org/10.1145/2959100.2959165
9. Ma, Y., Wang, S., Aggarwal, C.C., Yin, D., Tang, J.: Multi-dimensional graph convolutional networks. In: Proceedings of the 2019 SIAM International Conference on Data Mining, SIDM, pp. 657–665 (2019). https://doi.org/10.1137/1.9781611975673.74
10. Berg, R.V., Kipf, T., Welling, M.: Graph convolutional matrix completion. arXiv:1706.02263 (2017)
11. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the Neural Information Processing Systems, NIPS, pp. 1024–1034 (2017)
12. Wu, C., Ahmed, A.A., Beutel, A., Smola, A., Jing, H.: Recurrent recommender networks. In: Proceedings of the 10th ACM International Conference on Web Search and Data Mining, WSDM, pp. 495–503 (2017). https://doi.org/10.1145/3018661.3018689
13. Kipf, T., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the International Conference on Learning Representations, ICLR (2017)
14. Tang, J., et al.: Towards neural mixture recommender for long range dependent user sequences. In: Proceedings of the World Wide Web Conference, WWW, pp. 1782–1793 (2019). https://doi.org/10.1145/3308558.3313650
15. Kang, W., McAuley, J.J.: Self-attentive sequential recommendation. In: Proceedings of the International Conference on Data Mining, ICDM, pp. 197–206 (2018). https://doi.org/10.1109/icdm.2018.00035
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N.: Attention is all you need. In: Proceedings of the Neural Information Processing Systems, NIPS, pp. 5998–6008 (2017)
17. Fan, W., et al.: Graph neural networks for social recommendation. In: Proceedings of the World Wide Web Conference, WWW, pp. 417–426 (2019). https://doi.org/10.1145/3308558.3313488
18. Xu, W., Xu, Z., Zhao, B.: A graph kernel based item similarity measure for top-n recommendation. In: Proceedings of International Conference on Web Information Systems and Applications, WISA, pp. 684–689 (2019). https://doi.org/10.1007/978-3-030-30952-7_69