Pairwise-Based Hierarchical Gating Networks for Sequential Recommendation

Kexin Huang[1], Ye Du[1], Li Li[1(✉)], Jun Shen[2], and Geng Sun[2]

[1] School of Computer and Information Science, Southwest University, Chongqing, China
{huangkexin,duye99}@email.swu.edu.cn, lily@swu.edu.cn
[2] University of Wollongong, Wollongong, Australia
{jshen,jsun}@uow.edu.au

Abstract. The sequential pattern behind users' behaviors indicates the importance of exploring the transition relationships among adjacent items in next-item recommendation task. Most existing methods based on Markov Chains or deep learning architecture have demonstrated their superiority in sequential recommendation scenario, but they have not been well-studied at a range of problems: First, the influence strength of items that the user just access might be different since not all items are equally important for modeling user's preferences. Second, the user might assign various interests to certain parts of items, as what often attracts users is a specific feature or aspect of an item. Third, many methods ignore the complex item relations in user's previous actions. In this paper, we present a novel recommendation approach with gating mechanism and encoding module to address above problems. Specifically, the pair-wise encoding layer is first introduced to build 3-way tensor for modeling the relationships among items in user interact histories. We also apply two gating layers to filter useful information and capture user's short-term preference from aspect-level and item-level. We also follow the similar sprits to model user's long-term preference by integrating user latent embeddings. Empirical results on three public datasets show that our method achieves effective improvements over the state-of-the-art sequence-based models.

Keywords: Sequential recommendation · Collaborative filtering · Gating mechanism · Pairwise encoding

1 Introduction

Recommender system has become a popular way to alleviate information overload issue. Among the various recommendation methods, Collaborative filtering (CF) [1] is a most essential model to capture users' general preferences owing to its effectiveness and interpretability, but it fails to model the sequential dynamics in recommendation task. Leveraging users' behavior history sequences instead of ratings to predict their future behaviors has become increasingly popular in

G. Li et al. (Eds.): KSEM 2020, LNAI 12275, pp. 64–75, 2020.
https://doi.org/10.1007/978-3-030-55393-7_6

recent years [2,3]. This is because users access items in chronological order and the items that user will consume may be related to the items that he has just visited. To facilitates this task, a line of works convert users' historical actions into an action sequence order by operating timestamps [3–5].

Different from the conventional recommendation models, sequential recommendation methods usually based on Markov Chains (MCs) [2], which is a classic model that assume next action depends on previous actions and model the transition relationships between adjacent items for predicting user preferences. Although MCs-based models perform well in sparse scenarios, yet they cannot capture complex sequential dynamics. Another line of researches make use of deep neural networks (DNNs) to model both personalization and transitions based on item sequences, which outperform the MCs-based baselines. For example, Convolutional Neural Networks (CNNs) [6,7] have been introduced to capture user's short-term preferences, it adopts convolutional feature detectors to extract local patterns from item sequences by various sliding windows.

However, common neural methods regard sequence as a whole to calculate the impact on next item, which are difficult to gather relation features of different positions. Since a user may focus on one specific aspect of an item and pay different attention to various aspects of the same item. Furthermore, the item influence strength based on users' behaviors is diverse and dynamic, yet DNNs-based models fail to consider the specific aspect or feature of different items and ignore the item importance based on users' sequential actions.

In this paper, we also take user's sequences of recent interactions into account for sequential recommendation and follow the similar spirits [8] to apply Gate Convolutional Networks for modeling sequential dynamics for better recommendation. Specifically, we propose Hierarchical Pairwise Gating Model (HPGM) to effectively capture the sequential pattern then applying two gate linear units to model transition relationships and represent high-level features. For better relation extraction, we further devise pairwise encoding layer with concatenation which learn more meaningful and comprehensive representation of item sequence. We also conduct a series of experiments on serval benchmarks datasets. More importantly, experimental results show our model achieves better improvement over strong baselines.

2 Related Work

2.1 General Recommendation

General recommendation always focus on user's long-term and static preferences by modeling user's explicit feedbacks (e.g., ratings). Matrix Factorization (MF) [9] is the basis of many state-of-the-art methods such as [10], it seeks to uncover latent factors for representing users' preferences and items' properties from user-item rating matrix through the inner product operation. MF relies on user's explicit feedbacks but user's preferences also can be mined from implicit feedbacks (e.g., clicks, purchases, comments). The pair-wise methods [11] based on MF have been proposed to mining users' implicit actions and assume that user's

observed feedbacks should only be 'more preferable' than unobserved feedbacks then optimize the pairwise rankings of pairs.

Neighborhood-based and model-based methods also have been extended to tackle implicit feedbacks, a line of works are based on Item Similarity Matrix such as SLIM [12], FISM [13]. These methods calculate preference scores for a new item by measuring its similarities with previous items. Recently, various deep learning techniques have been introduced to extract item features from the description of items such as images and texts by neural network in recommendation task [14].

2.2 Sequential Recommendation

For sequential recommendation task, Markov Chains (MCs) is an effective method to model sequential dynamics from successive items. Factorized Personalized Markov Chains (FPMC) [2] is a classic sequential recommendation model that combines MF and factorized MCs to model user preference and sequential patterns simultaneously. Hierarchical Representation Model (HRM) [4] extends the FPMC by introducing aggregation operations like max-pooling to model more complex. He et al's method (TransRec) [3] models the third-order interactions between sequential items by combining with metric embedding approaches.

Besides, another line of works model user sequences via deep learning techniques and show effective performance in sequential recommendation task [15]. Convolutional Sequence embedding (Caser) [6] captures sequential patterns and transitions from previous item sequence by convolutional operation with various filters. Another popular method RNN is also used to model user's sequential interactions because RNN is good at capturing transition patterns in sequence [16,17]. Attention Mechanisms have been incorporated into next item recommendation to model complex transitions for better recommendation [18]. Self-attention based sequential model (SASRec) [19] relies on Transformer instead of any recurrent and convolutional operations, it models the entire user sequence to capture user's long-term and short-term preferences then make predictions on few actions.

3 Proposed Methodology

The objective of our task is to predict next behaviors of users depending on previous chronological actions. We use $\mathcal{U}$ and $\mathcal{I}$ to present user set and item set (respectively) in sequential recommendation scenario. Given user u's action sequence $\mathcal{S}^u = (\mathcal{S}_1^u, \mathcal{S}_2^u, \cdots, \mathcal{S}_{|\mathcal{S}^u|}^u)$, where $\mathcal{S}_t^u \in \mathcal{I}$ denotes user u ever interacted with the item at time step t. To train the network, we extract every L successive items $(\mathcal{S}_1^u, \mathcal{S}_2^u, \cdots, \mathcal{S}_L^u)$ of each user $u \in \mathcal{U}$ as the input, its expected output as the next T items from the same sequence: $(\mathcal{S}_{L+1}^u, \mathcal{S}_{L+2}^u, \cdots, \mathcal{S}_{L+T}^u)$. In this section, we introduce our model via an embedding layer, pairwise encoding layer, hierarchical gating layer and prediction layer. The detailed network architecture is showed in Fig. 1.

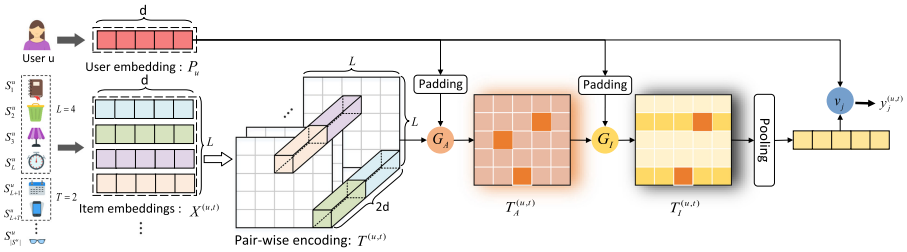


Fig. 1. The detail network architecture of **HPGM**. Previous successive item embeddings are transmitted into the pairwise encoding layer, the output and user embedding after pooling are passed into the hierarchical gating layer (G_A, G_I) and then predict the next item by combining the original user embedding and the sequence embedding after pooling operation.

3.1 Embedding Layer

Let $E_i \in \mathbb{R}^d$ be the item embedding corresponding to the i-th item in the item sequence, where d is the latent dimensionality. The embedding look-up operation retrieves previous L items' embeddings and stack them together to form the input matrix $X^{(u,t)} \in \mathbb{R}^{L \times d}$ for user u at time step t. Along with item embedding, we also represent user features in latent space with user embedding $P_u \in \mathbb{R}^d$.

3.2 Pairwise Encoding Layer

In order to capture intricate item relations among a specific item sequence and improve the flexibility of the model, we use pair-wise encoding layer to build a sequential tensor $T^{(u,t)} \in \mathbb{R}^{L \times L \times 2d}$ to store various item relationships. $T^{(u,t)}$ is composed by the item pair(i, j) of item subsequence, which concatenate the embedding of item i and j. The encoded 3-way tensor is similar to "image feature map" in the CNN-based model for computer vision tasks, so $T^{(u,t)}$ can replace the sequential item embedding $X^{(u,t)}$ as the input to downstream layers. Note we padding the user embedding with "1" and generate a user tensor $\hat{P}_u$ with same dimensions of $T^{(u,t)}$ for feeding the user embedding into the subsequent layers.

3.3 Hierarchical Gating Layer

Original GLU integrate convolution operation and simplified gating mechanism to make predictions [20], motivated by the gated linear unit (GLU) utilized on recommendation task [8], we also adopt similar spirits to model sequence dynamics. GLU control what information should be propagated for predicting next item, so we can select specific aspect/feature of item and particular item that is related to future items.

Aspect-Level Gating Layer. A user generally decides whether to interact with the item by looking for the specific attractive aspects of the item. Therefore, we modify the GLU to capture sequence pattern based on user-specific preference. The convolution operation is replaced by inner product to reduce the parameters of the model and the user's aspect-level interest can be generated by:

$$T_A^{(u,t)} = T^{(u,t)} * \sigma(W_1 \cdot T^{(u,t)} + W_2 \cdot \hat{P}_u) \tag{1}$$

where $*$ is the element-wise multiplication, $\cdot$ represents inner product operation, W_1, $W_2 \in \mathbb{R}^{1\times 2d\times 2d}$ and $b \in \mathbb{R}^{1\times 1\times 2d}$ are the corresponding 3-way weight terms and the bias term, $\sigma(\cdot)$ denotes the sigmoid function. And the aspect-specific information can be propagated to the next layer by the aspect-level gating layer.

Item-Level Gating Layer. Users will assign higher weight attention to a particular item in real life. Exiting models ignore the item importance in modeling users' short-term preferences and attention mechanism is a success way to capture item-level interest. In this paper, we also adopt an item-level gating layer to achieve the same or even better performance. And the results after this layer can be calculated as:

$$T_I^{(u,t)} = T_I^{(u,t)} * \sigma(W_3 \cdot T_A^{(u,t)} + W_4 \cdot \hat{P}_u) \tag{2}$$

where $W_3 \in \mathbb{R}^{1\times 1\times 2d}$, $W_4 \in \mathbb{R}^{1\times L\times 2d}$ are learnable parameters. By performing aspect-level and item-level gating module operations on item embedding, our model selects informational items and their specific aspects, meanwhile eliminates irrelevant features and items. Then we apply average pooling on the sequence embedding after item-level gating layer to make aggregation by accumulating the informative parts:

$$\hat{E}^{(u,t)} = average\left\{T_I^{(u,t)}\right\} \tag{3}$$

3.4 Prediction Layer

After computing user's short-term preference by preceding operation, we induce an implicit user embedding P_u to capture user's general preferences then we employ the conventional latent factor model (matrix factorization) to generate prediction score as follows:

$$y_j^{(u,t)} = \hat{E}^{(u,t)} v_j + P_u v_j \tag{4}$$

where $y_j^{(u,t)}$ can be interpreted as the probability of how likely user u will interact with item j at time step t and v_j denotes the item embedding. Note we adopt the full-connected layer to reduce the high-dimension before prediction.

3.5 Network Training

To train the network, we adopt the binary Bayesian Personalized Ranking loss [11] as the objective function:

$$L = \sum_{(u,i,j)\in\mathcal{D}} -ln\sigma(y_i^u - y_j^u) + \lambda_\Theta(\|\Theta\|^2) \tag{5}$$

where $\Theta = \{X, P_u, W_1, W_2, W_3, W_4, b\}$ denotes the model parameters, which are learned by minimizing the objective function on training set. Note we use some tricks to learn these 3-way parameters by PyTorch and their dimensions are derived from experiments. λ_Θ is the regularization parameter and $\sigma(x) = 1/(1+e^{-x})$, $\mathcal{D}$ is the set of training triplets:

$$\left\{(u,i,j)\,|u \in \mathcal{U} \wedge i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I}_u^-\right\} \tag{6}$$

we also randomly generate one negative item j from a candidate set of each user in each time step t, the candidate set of each user is defined by $\{j \in \mathcal{I}^-|\mathcal{I}^- = \mathcal{I} - \mathcal{S}^u\}$ and the Adam Optimizer [21] is used to optimize the network.

4 Experiments

In order to evaluate our model, we experiment with various baselines on three large-scale real-world datasets. The datasets cover different domains and sparsity. All the datasets and code we used are available online.

4.1 Datasets

We evaluate our model on three real-world dataset and these datasets vary greatly in domain, variability, sparsity and platform:

Amazon[1]. This dataset is collected from *Amazon.com* that contains large corpora of products ratings, reviews, timestamps as well as multiple types of related items. In this work, we choose the "CDs" category to evaluate the quantitative performance of the proposed model.

MovieLens[2]. MovieLens is created by the Grouplens research group from *Movielen.com*, which allows users to submit ratings and reviews for movies they have watched.

GoodReads[3]. A new dataset introduced in [22], comprising a large number of users, books and reviews with various genres. This dataset is crawled from

[1] http://jmcauley.ucsd.edu/data/amazon/.
[2] https://grouplens.org/datasets/movielens/.
[3] https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home.

Goodreads, a large online book review website. In this paper, we adopt the genres of Comics to evaluate the proposed model.

For each of the above datasets, we follow the same preprocessing procedure from [6]. We converted star-ratings to implicit feedback and use timestamps to determine the sequence order of users' actions. In addition, we discard users and items with less than 5 related actions. We also partition the sequence $\mathcal{S}^u$ for each user u into three parts: (1) the first 70% of actions in $\mathcal{S}^u$ as the training set. (2) the second 10% of actions for validation. (3) the remaining 20% of actions are used as a test set to evaluate performance of the model. Statistics of each dataset after pre-processing are shown in Table 1.

Table 1. Dataset statistics.

Dataset	#users	#items	#actions	Avg. #actions /user	Avg. #actions /item
Amazon CDs	17.0K	35.1K	0.47M	27.69	13.44
MovieLens	129.9K	13.6K	9.9M	76.43	726.89
GoodReads Comics	34.4K	33.1K	2.4M	70.00	72.80

4.2 Comparison Methods

We contain three groups of recommendation baselines to show the effective of HPGM. The first group are general recommendation models which only take user feedbacks into account instead of considering user's sequential behaviors.

- **PopRec**: PopRec ranks items according to the order of their overall popularity which decided by the number of the interactions.
- **Bayesian Personalized Ranking (BPR-MF)** [11]: This model combines matrix factorization and learning personalized ranking from implicit feedback by Bayesian Personalized Ranking.

The next group of the methods models the sequence of user actions to explore user's preference in sequential recommendation:

- **Factorized Markov Chains (FMC)** [2]: FMC factorizes the first-order Markov transition matrix to capture 'global' sequential pattern but it ignores the personalized user interaction.
- **Factorized Personalized Markov Chains (FPMC)** [2]: FPMC combines the matrix factorization and factorized Markov Chains as its recommender and it captures item-to-item transition and users' long-term preference simultaneously.

The final group includes methods which consider serval previously visited items to make predictions by deep-learning technique.

- **Convolutional Sequence Embeddings (Caser)** [6]: Caser captures sequential dynamic by convolutional operations on embedding matrix with length L.
- **GRU4Rec** [23]: This model treats users' action sequence as a session and utilizes RNNs to model user feedback sequences for session-based recommendation.
- **GRU4Rec$^+$** [24]: GRU4Rec$^+$ extends the GRU4Rec method by applying a different loss function and sampling strategy and achieve great sequential recommendation performance.

4.3 Evaluation Metrics

In order to evaluate performance of sequential recommendation, we adopt two common Top-N metrics Recall@N and NDCG@N. Recall@N measure Top-N recommendation performance by counting the proportion of times that the ground-truth next item is among the top N items and NDCG@N is a position-aware metric that distribute high weights on the higher positions. Here N is set from $\{5, 10, 15, 20\}$.

4.4 Implementation Details

The parameters of baselines are initialized as corresponding number in original paper. The latent dimension d is tested in $\{10, 20, 30, 40, 50\}$ and the learning rate for all models are tuned amongst $\{0.001, 0.005, 0.01, 0.02, 0.05\}$. We tune the batch size in $\{16, 32, 64, 128\}$ and margin λ_Θ is tuned in $\{0.001, 0.005, 0.01, 0.02\}$. After tuning processing on validation set, the learning rate is set to 0.001, $d = 50$, $\lambda_\Theta = 0.001$ and the batch size is 256. We also follow the same setting in: the Markov order L is 5 and predict the future $T = 3$ items. All experiments are implemented with PyTorch[4].

4.5 Recommendation Performance

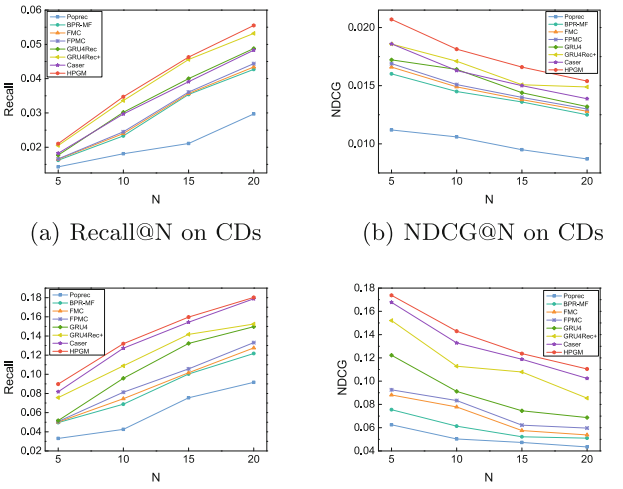
Overall performance results of HPGM and baselines are summarized in Table 2 and Fig. 2, which clearly illustrate that our model obtains promising performance in terms of Recall and NDCG for all reported values in sequential recommendation task. We can gain the following observations:

The performance of BPR-MF is better than PopRec but is not as good as FMC, which demonstrates that local adjacent sequential information plays an vital role under the typical sequential recommendation setting. Compared to conventional sequential-based models (FMC and FPMC), we find that item-to-item relations is necessary to comprehend user's sequential actions. Furthermore, the performance results show that our proposed model can effectively capture item relationships and sequential dynamics in real-world datasets.

[4] https://pytorch.org/.

Table 2. Performance comparison with baselines on three datasets and the best results highlight in bold (Higher is better). The improvement is calculated by the best performance of baselines and our method.

Dataset	*Amazon-CDs*		*MovieLens*		*GoodReads-Comics*	
Metrics	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
PopRec	0.0181	0.0095	0.0560	0.0487	0.0426	0.0503
BPR-MF	0.0233	0.0145	0.0774	0.0685	0.0688	0.0613
FMC	0.0240	0.0149	0.0819	0.0724	0.0745	0.778
FPMC	0.0245	0.0151	0.0847	0.0751	0.0813	0.0833
GRU4Rec	0.0302	0.0164	0.0924	0.0815	0.0958	0.0912
GRU4Rec$^+$	0.0336	0.0171	0.1004	0.0946	0.1088	0.1128
Caser	0.0297	0.0163	0.1139	0.1016	0.1273	0.1329
HEPG	**0.0347**	**0.0181**	**0.1150**	**0.1087**	**0.1320**	**0.1430**
%Improv.	3.36	6.08	0.97	7.01	3.67	7.56

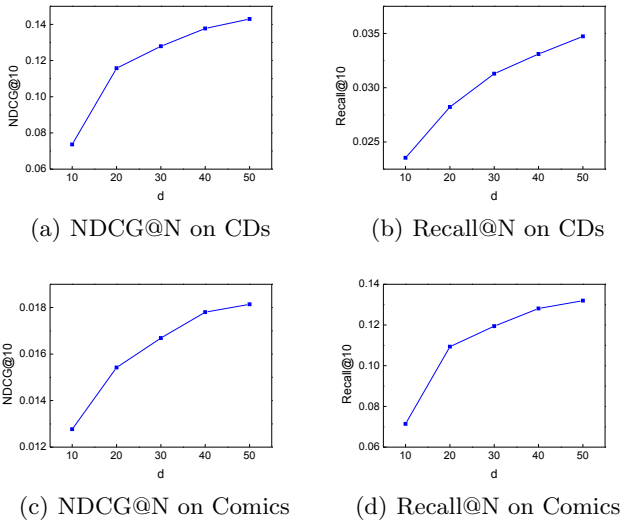


(a) Recall@N on CDs (b) NDCG@N on CDs

(c) Recall@N on Comics (d) NDCG@N on Comics

Fig. 2. Ranking performance (NDCG and Recall) with baselines on Amazon-CDs and GoodReads-Comics.

Another observation is sequential methods GRU4Rec and Caser based on neural network achieve better performance than conventional sequential recommendation model such as FPMC. We can conclude that neural network is suitable to model the complex transition between previous feedbacks and future behaviors of the user. Since baseline models have a lot of limitation, Caser only considers group-level influence by adopting CNN with horizontal and vertical filters but ignores the specific aspect-level influence of successive items.



(a) NDCG@N on CDs (b) Recall@N on CDs

(c) NDCG@N on Comics (d) Recall@N on Comics

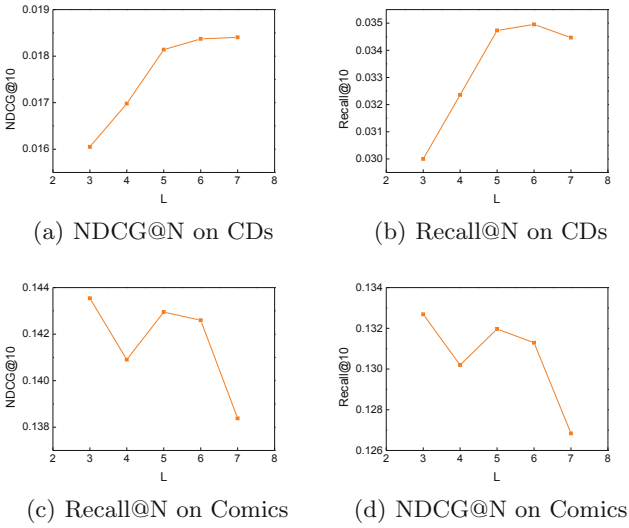
Fig. 3. Performance change with different dimension of embeddings d on Amazon-CDs and GoodReads-Comics.

In a word, our method can beat baselines with ground-truth ranking and shows effectiveness of our model on item relation, sequential dynamics and user's general preferences.

4.6 Influence of Hyper-parameters

In this subsection, we also analyze the effect of two key hyper-parameters: the latent dimensionality d and the length of successive items L. Figure 3 shows the effect of dimension d by evaluating with NDCG@10 and Recall@10 of all methods varying from 10 to 50 on Amazons-CDs and GoodReads-Comics. We also can conclude that our model typically benefits from lager dimension of item embeddings. Since small latent dimension cannot express the latent feature completely and with the increase of d, the model can achieve better performance on the real-world datasets.

Previous analysis can demonstrate that modeling sequence patterns are crucial for next-item recommendation, hence the length of sequence is a significant factor to determine model's performance. We also study the influence of different length of successive items L and Fig. 4 shows that the model does not consistently benefit from increasing L and a large L may lead to worse results since higher L may introduce more useless information. In most cases, $L = 5$ achieve better performance on the two datasets.



(a) NDCG@N on CDs (b) Recall@N on CDs

(c) Recall@N on Comics (d) NDCG@N on Comics

Fig. 4. The performance of HPGM with varying L on Amazon-CDs and GoodReads-Comics.

5 Conclusion

In this paper, we present a novel recommendation approach with gating mechanism to learn personalized user and item representations from user's sequential actions and generating prediction score by aggregating user's long-term and short-term preferences. Specifically, in order to model item relations in user behaviors, we apply pair-wise encoding layer to encode a sequence of item embedding into a pairwise tensor. Moreover, we build a hierarchical gating layer to model aspect-level and item-level influence among items to capture latent preferences of the user. We also conduct extensive experiments on multiple large-scale datasets and the empirical results show that our model outperforms state-of-the-art baselines. In the future, we plan to extend the model by exploring sequential patterns and make predictions from various types of context information.

Acknowledgement. This research was supported by NSFC (Grants No. 61877051), and Natural Science Foundation Project of CQ, China (Grants No. cstc2018jscx-msyb1042, and cstc2018jscx-msybX0273). Li Li is the corresponding author for the paper.

References

1. Ekstrand, M.D., Riedl, J.T., Konstan, J.A., et al.: Collaborative filtering recommender systems. Found. Trends® Hum.-Comput. Interact. **4**(2), 81–173 (2011)
2. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the WWW, pp. 811–820. ACM (2010)

3. He, R., Kang, W.-C., McAuley, J.: Translation-based recommendation. In: Proceedings of the RecSys, pp. 161–169. ACM (2017)
4. Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., Cheng, X.: Learning hierarchical representation model for next basket recommendation. In: Proceedings of the SIGIR, pp. 403–412. ACM (2015)
5. Yu, L., Zhang, C., Liang, S., Zhang, X.: Multi-order attentive ranking model for sequential recommendation. In: Proceedings of the AAAI (2019)
6. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the WSDM, pp. 565–573. ACM (2018)
7. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: Proceedings of the WSDM, pp. 582–590. ACM (2019)
8. Ma, C., Kang, P., Liu, X.: Hierarchical gating networks for sequential recommendation. In: Proceedings of the KDD, pp. 825–833 (2019)
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **8**, 30–37 (2009)
10. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: Proceedings of the WWW, pp. 173–182. ACM (2017)
11. Rendle, S., Freudenthaler, C., Gantner, Z.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
12. Ning, X., Karypis, G.: Slim: sparse linear methods for top-n recommender systems. In: Proceedings of the ICDM, pp. 497–506. IEEE (2011)
13. Kabbur, S., Ning, X., Karypis, G.: FISM: factored item similarity models for top-n recommender systems. In: Proceedings of the KDD, pp. 659–667. ACM (2013)
14. Kang, W.-C., Fang, C., Wang, Z., McAuley, J.: Visually-aware fashion recommendation and design with generative image models. In: Proceedings of the ICDM, pp. 207–216. IEEE (2017)
15. Yan, A., Cheng, S., Kang, W.-C., Wan, M., McAuley, J.: CosRec: 2D convolutional neural networks for sequential recommendation. arXiv preprint arXiv:1908.09972 (2019)
16. Xu, C., Zhao, P., Liu, Y.: Recurrent convolutional neural network for sequential recommendation. In: Proceedings of the WWW, pp. 3398–3404. ACM (2019)
17. Yu, Z., Lian, J., Mahmoody, A., Liu, G., Xie, X.: Adaptive user modeling with long and short-term preferences for personalized recommendation. In: Proceedings of the IJCAI, pp. 4213–4219, July 2019
18. Zhou, C., et al.: ATRank: an attention-based user behavior modeling framework for recommendation. In: Proceedings of the AAAI (2018)
19. Kang, W.-C., McAuley, J.: Self-attentive sequential recommendation. In: Proceedings of the ICDM, pp. 197–206. IEEE (2018)
20. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Proceedings of the ICML, vol. 70, pp. 933–941. JMLR.org (2017)
21. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the ICLR (2015)
22. Wan, M., McAuley, J.: Item recommendation on monotonic behavior chains. In: Proceedings of the RecSys, pp. 86–94. ACM (2018)
23. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)
24. Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations. In: Proceedings of the CIKM, pp. 843–852. ACM (2018)