Learning Shared Vertex Representation in Heterogeneous Graphs with Convolutional Networks for Recommendation

Yanan Xu , **Yanmin Zhu**[*] , **Yanyan Shen** and **Jiadi Yu**
Shanghai Jiao Tong University, Shanghai, China
{xuyanan2015, yzhu, shenyy, jiadiyu}@sjtu.edu.cn

Abstract

Collaborative Filtering (CF) is among the most successful techniques in recommendation tasks. Recent works have shown a boost of performance of CF when introducing the pairwise relationships between users and items or among items (users) using interaction data. However, these works usually only utilize one kind of information, i.e., user preference in a user-item interaction matrix or item dependency in interaction sequences which can limit the recommendation performance. In this paper, we propose to mine three kinds of information (user preference, item dependency, and user similarity on behaviors) by converting interaction sequence data into multiple graphs (i.e., a user-item graph, an item-item graph, and a user-subseq graph). We design a novel graph convolutional network (PGCN) to learn shared representations of users and items with the three heterogeneous graphs. In our approach, a neighbor pooling and a convolution operation are designed to aggregate features of neighbors. Extensive experiments on two real-world datasets demonstrate that our graph convolution approaches outperform various competitive methods in terms of two metrics, and the heterogeneous graphs are proved effective for improving recommendation performance.

1 Introduction

With the explosive growth of information, recommender systems have become indispensable components for many online services, such as e-commerce and online entertainment. Two major branches of recommendation algorithms are collaborative Filtering (CF) methods [Koren and Bell, 2015] and content-based recommender systems [Wang and Wang, 2014]. Collaborative filtering methods, especially matrix factorization [Koren *et al.*, 2009], have been proved to be effective by using interactions (e.g., ratings and clicks) to predict user preference on items.

Recent works have shown a boost of performance of CF methods by exploiting relationship between users and items or relationships among users (items) [Cheng *et al.*, 2017; Kabbur *et al.*, 2013; He *et al.*, 2018]. By utilizing interaction data, two kinds of approaches have been proposed to construct the relationships. One way is to store preference relationships (i.e., interactions) between users and items in a user-item matrix. A user's profile is built with her historically interacted items (neighbors) [Kabbur *et al.*, 2013; He *et al.*, 2018]. Another approach is to convert original interaction records of each user to interaction sequences of items according to timestamps. It is believed that item similarity/dependency can be mined from interaction sequences, since the next item may depend on the previous items. The item dependency is directly used to predict the interactions in the future [Feng *et al.*, 2015; Song *et al.*, 2016; Yoo *et al.*, 2017] or is used as item similarity constraints to enhance CF methods [Cheng *et al.*, 2017]. However, these works usually only utilize one kind of relationships, i.e., user preference or item dependency, which may limit the performance of recommender systems.

Besides the item dependency, we argue that interaction sequences can also reflect user similarity on behaviors which has not been well exploited. The number of shared subsequences of interactions between two users can indicate their similarity on behaviors. For example, shared sequences of played songs can reveal similar tastes of users as these songs may be of the same album, artist, or genre [Cheng *et al.*, 2017]. Repeated shopping sequences may indicate a user's profile such as occupation. A student may have many shopping sequences like $pen, pencil, eraser$.

In this paper, we aim to mine three kinds of information, i.e., user preference, item dependence, and user similarity on behaviors, from interaction data. But there are three challenges that need to be addressed. First, a suitable data structure should be designed to retain the three kinds of relationships. The traditional user-item interaction matrix can only record interactions between users and items which indicate user preference and omits the other two kinds of information. Second, neighbors (users or items) defined by the relationships should be selected carefully as they may bring noises. Third, for a user or an item, we need to consider how to model the effect of its neighbors.

To solve these challenges, we first convert interaction sequences into three graphs including a user-item graph, an

[*]Corresponding author

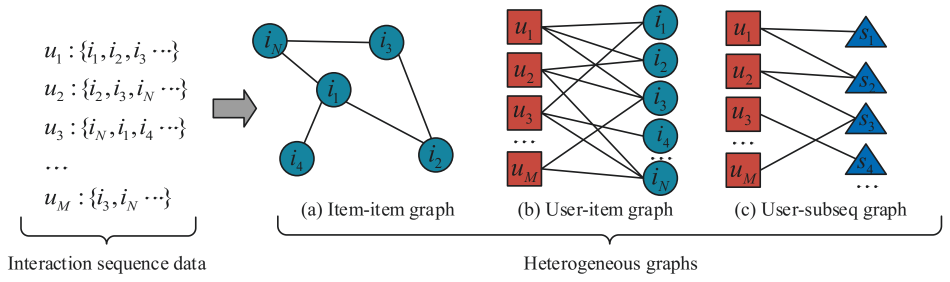

$u_1 : \{i_1, i_2, i_3 \cdots\}$
$u_2 : \{i_2, i_3, i_N \cdots\}$
$u_3 : \{i_N, i_1, i_4 \cdots\}$
$\cdots$
$u_M : \{i_3, i_N \cdots\}$
Interaction sequence data
(a) Item-item graph
(b) User-item graph
(c) User-subseq graph
Heterogeneous graphs

Figure 1: Illustration of converting interaction sequences to heterogeneous graphs.

item-item graph, and a user-subseq graph to indicate the user preference over items, item dependency, and user similarity, respectively. In the graphs, neighbors are selected according to paths between vertices (path type and distance). Next, we propose a graph convolutional network named PGCN to learn the three kinds of information with the graphs and produce shared representations for users and items. The traditional convolution operation can only be applied on grid-structured input. But the graphs are in non-Euclidean domains and the convolution cannot handle the varied number of neighbors [Hechtlinger *et al.*, 2017]. To address this issue, we design a neighbor pooling operation to cluster neighbors according to the path type and distance. For each target vertex, the pooling operation produces a fixed number of virtual vertices by aggregating features of its neighbors with attention mechanism. We perform a convolution operation on the virtual vertices and generate new latent vectors for each vertex (users or items). Then, our neural network outputs shared representations for users and items in heterogenous graphs. At last, the recommendation problem is formulated as an edge prediction task based on the representations.

In summary, our main contributions are as follows.

- We design three heterogeneous graphs for mining user preference, item dependency and user similarity information contained by interaction sequences for recommendation tasks.
- We propose a graph convolutional network, PGCN, to learn shared representations for users and items with the heterogeneous graphs. We offer a novel pooling operation to deal with a varied number of neighbors and design a graph convolution operation for aggregating features of neighbors and retaining the locality in graphs.
- Our approach outperforms several state-of-the-art methods in terms of hit ratio and NDCG on two real-world datasets. The experiments also prove that the heterogeneous graphs are effective for improving the recommendation performance.

2 Preliminary

2.1 Learning from Sequence Data

We first present several fundamental definitions.

Definition 1 (Interaction Sequence) *For each user, his/her interaction records (i.e., $< user, item, time >$) can be sorted by timestamps and form sequences as shown in Figure 1.*

Definition 2 (N-item subsequence) *An n-item subsequence is a contiguous sequence of n items from a given interaction sequence of a user. For example, a user has a shopping sequence {pen, pencil, eraser}. Its 2-item subsequences are "pen, pencil" and "pencil, eraser". We use subsequences to denote all n-item subsequences.*

We convert interaction sequences into following three graphs (Figure 1).

Definition 3 (User-item Interaction Graph) *The user-item interaction graph $G^R = (V^U \cup V^I, E^R)$ records interactions between users and items, where V^U and V^I denote sets of users and items, respectively, and $|V^U| = M$, $|V^I| = N$. $E^R \subseteq V^U \times V^I$ is a set of edges. The weight of edge $e_{u,i}$ denotes the number of interactions between u and i.*

Definition 4 (Item-item Graph) *The item-item graph, i.e., $G^I = (V^I, E^I)$, is designed to retain the dependency between items. The V^I is the set of items. The weight of edge $e_{i,j}$ in $E^I \subseteq V^I \times V^I$ is the number of co-occurrence of item i and j (two adjacent items) in interaction sequences.*

Definition 5 (User-subseq Graph) *The user-subseq graph $G^U = (V^U \cup V^G, E^U)$ is utilized to record interactions between users and subsequences and indicates user similarity on behaviors. V^G is the set of n-item subsequences and $E^U \subseteq V^U \times V^G$ is the set of edges in G^U.*

At last, we use G to denote the three graphs, i.e., $G = G^R \cup G^I \cup G^U$. The three graphs can also be represented by adjacency matrices, i.e., $A^R \in \mathbb{R}^{(M+N)\times(M+N)}$, $A^I \in \mathbb{R}^{N\times N}$, $A^U \in \mathbb{R}^{(M+O)\times(M+O)}$. The elements in adjacency matrices record weights of edges in the corresponding graphs.

If there exists a path from vertex i to vertex j in a graph, we define the *path type* $T(i,j)$ with the types of i and j. The number of edges in the path is defined as *distance* $D(i,j)$ between the two vertices. For example, a path $user_1 \rightarrow item_1 \rightarrow user_3$ in graph G^R, its path type will be u-u (user to user) and distance will be 2.

	Distance=1			Distance=2		
Graphs	G^I	G^R	G^U	G^I	G^R	G^U
User	/	$u-i$	$u-s$	/	$u-u$	$u-u$
Item	$i-i$	$i-u$	/	$i-i$	$i-i$	/
Subseq	/	/	$s-u$	/	/	$s-g$

Table 1: Path types in the three kinds of graphs. User, item and subsequence are denoted by u, i and s, respectively.

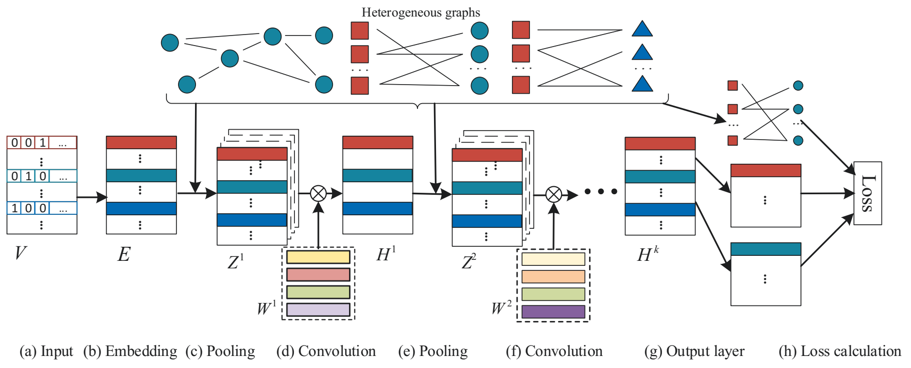


Figure 2: Structure of PGCN. V and E are one-hot representation and embedding matrices of vertexes, respectively. Zs and Hs are feature tensors of virtual vertexes after neighbor pooling and real vertexes after convolution, respectively. Ws are weight matrices.

Definition 6 (Neighbors) *Neighbors of one vertex in heterogeneous graphs are chosen by two factors, i.e.,* path type *and* distance. *We define neighbors of a vertex i as $N_{lt}(i) = \{j|D(i,j) = l, T(i,j) = t\}$ where l is distance and t is path type. In addition, we define each vertex as its neighbor with distance equal to* 0, *i.e., $N_{0,T(i)}(i) = \{i\}$. Neighbors of each kind of vertex are shown in Table 1.*

2.2 Recommendation Problem Formulation

In our work, interactions between users and items are represented by edges between vertices in graphs. The edges reflect users' preference on items. Therefore, the recommendation problem can be cast as an edge prediction task. More precisely, given the heterogeneous graphs G, we aim to predict the probability of existence of edge $e_{u,i}$ between vertex u and i, i.e., $p(e_{u,i}|G)$.

3 Learning Shared Vertex Representations

In this section, we propose a new graph convolutional neural network, Path conditioned Graph Convolutional Network (PGCN) as shown in Figure 2. In our approach, we represent vertex features and adjacency information with a matrix and heterogeneous graphs, respectively. In what follows, we introduce details of our approach layer by layer.

3.1 Input and Embedding Layer

As shown in Figure 2(a), each vertex (including users, items and n-item subsequences) is denoted by a one-hot vector v_i indicating its ID. It should be noticed that properties or contents of vertices can also be concatenated with one-hot vectors to alleviate the data sparsity problem. As we aim to study the effect of convolution on a graph with information of neighbors, only one-hot vectors are employed for simplicity. We use matrix $V \in \mathbb{R}^{d_0 \times d_0}$ ($d_0 = M + N + O$) to store one-hot vectors of all vertices.

The one-hot vectors are transformed into embeddings of low dimensions with $H^0 = VE$ and $E \in \mathbb{R}^{d_0 \times d_1}$. The embedding of vertex i is denoted by h_i^0. The three graphs share embeddings and representations of users and items.

3.2 Neighbor Pooling

Now, we need to define a convolution operation to reserve the local connectivity of graphs. The traditional convolution operation cannot handle the varying number of neighbors for each vertex. However, another widely-used operation, i.e., pooling, can solve the problem. We employ the pooling operation to aggregate features of neighbors and produce a fixed number of virtual vertices. In previous research, pooling layers usually come after convolution layers. But the two operations are actually independent of each other. We can conduct a pooling operation before a convolution layer.

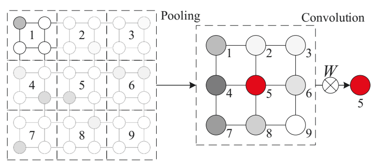
In Figure 3(a), the pooling operation is performed on 9 regions of an image with four nodes (pixels) in each of them. The regions are constructed with nodes according to their direction and distance to region 5. After pooling, a convolution operation is performed to produce the representation for node 5. Similarly, in graphs, we select neighbors by *path type* and *distance* to target vertices, i.e., $N_{lt}(i)$, which is shown as Figure 3(b).

We introduce the processing details of neighbor pooling. First, neighbors are clustered by the path type and distance to a target vertex. Then, for each cluster, a neighbor pooling is performed and produces a virtual vertex. The equation for pooling is as below.

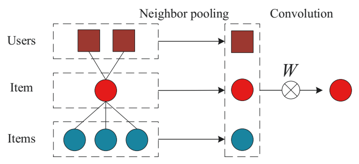
$$z_{lt}^i = \sum_{j \in N_{lt}(i)} \alpha_{ij} h_j^{k-1}, \quad (1)$$

where $N_{lt}(i)$ is the set of neighbors with distance l and path type t for vertex i. h_j^{k-1} is the representation of vertex j in layer $k-1$. α_{ij} is the weight for determining the importance of j for i. z_{lt}^i denotes one virtual vertex aggregated with neighbors N_{lt} of vertex i.

We provide two ways to calculate the weight α_{ij}, i.e., with transition probability or attention. We take the graph G^R as an example. For G^R, we have the adjacency matrix A^R and define D as a diagonal matrix where $D_{ii} = \sum_j A_{ij}^R$. Then the transition matrix for one hop can be obtained by $S^{(1)} = D^{-1}A^R$. Based on the transition matrix $S^{(1)}$, we can obtain a transition matrix for two hops with $A' = S^{(1)}S^{(1)}$. We set



(a) Pooling and convolution on an image



(b) Pooling and convolution on a graph

Figure 3: Illustration of pooling and convolution operations (dashed line box) on an image or a graph. Ws are weights for convolution.

$A'_{ii} = 0$ to remove the transition to the vertex itself. The matrix is normalized with $S^{(2)} = E^{-1}A'$ and E is a diagonal matrix where $E_{ii} = \sum_j A'_{ij}$. We set α_{ij} to $S^{(1)}_{ij}$ and $S^{(2)}_{ij}$ for neighbors with distance equal to 1 and 2, respectively.

The weight α_{ij} can also be computed with the attention mechanism according to the following equation [Velickovic *et al.*, 2017; He *et al.*, 2018].

$$\alpha_{ij} = \frac{exp(f(h_i^{k-1}, h_j^{k-1}))}{\sum_{j' \in N_{lt}(i)} exp(f(h_i^{k-1}, h_{j'}^{k-1}))}, \qquad (2)$$

where $f(\cdot)$ is a fully connected neural network with two hidden layers.

We use term “PGCN-C” and “PGCN-A” to denote our approaches that use constant transition probability and attention mechanism, respectively.

3.3 Convolution

Convolution is conducted on the virtual vertices and generates new representations of vertices:

$$h_i^k = g(\sum_{l=0}^{L} \sum_t W_{lt}^k z_{lt}^i + b_{lt}^k), \qquad (3)$$

where z_{lt}^i is the feature vector of a virtual vertex with distance l and path type t for vertex i. L determines the size of filters, i.e., how many neighbors are considered. For example, if $L = 1$, neighbors within 1 hops will be utilized. In this paper, we only consider neighbors within 2 hops. $W_{lt}^k \in \mathbb{R}^{d_k \times d_{k-1}}$ and $b_{lt}^k \in \mathbb{R}^{d_k}$ are weights and bias for convolutional filters in layer k. All vertices in graphs share parameters W_{lt}^k and b_{lt}^k for the same kinds of virtual vertices. $g(\cdot)$ is an activation function.

With several neighbor pooling and convolution layers, we can output a representation h_i for each vertex.

3.4 Model Training

With the representations of users and items, we can compute the probability of the existence of each edge between users and items with $\hat{p}(e_{u,i}|G) = \sigma({h_u}^T h_i)$. h_u and h_i are representations of user u and item i, respectively. The interaction function (inner product) between h_u and h_i can also be replaced by a fully connected neural network [He *et al.*, 2017]. For the space limitation, we leave this to be studied in the future. σ is the sigmoid function. $\hat{p}(e_{u,i}|G)$, $\hat{p}_{u,i}$ for short, denotes the predicted probability of existence of edge $e_{u,i}$. If a user has interacted with an item, the ground truth is set to 1, i.e., $p_{u,i} = 1$. Otherwise, $p_{u,i} = 0$. The loss for optimization can be defined with a cross-entropy function.

$$\mathcal{L} = -\left(\sum_{(u,i)\in\mathcal{Y}} p_{u,i} \log \hat{p}_{u,i} + (1 - p_{u,i}) \log(1 - \hat{p}_{u,i}) \right) + \lambda\omega(\theta), \quad (4)$$

where $\mathcal{Y} = \mathcal{Y}^+ \cup \mathcal{Y}^-$. $\mathcal{Y}^+$ denotes the set of existing edges and $\mathcal{Y}^-$ is a set of unobserved interactions between users and items. Given one interaction (u, i), negative instances (u, j) are sampled from items that user u has not interacted with. $\omega(\theta)$ is the regularization term and λ is the weight of it.

4 Experiments

In the experiments, we aim to answer the following three questions.

RQ1: Do our approaches outperform other methods?

RQ2: Do the heterogeneous graphs provide valuable information beyond the user preference on items?

RQ3: How do hyper-parameters affect the performance?

4.1 Experimental Settings

Dataset	# user	# item	# interaction
MovieLens	943	1,682	100,000
Retailrocket	1,407,580	417,053	2,664,312

Table 2: Statistics of datasets

Datasets. In the experiments, we utilize two publicly available datasets including MovieLens-100K[1] and Retailrocket[2]. Their statistic information is summarized in Table 2. Their descriptions are as follows:

1. MovieLens dataset collected ratings of movies (1-5 star) rated by users. We treated ratings as weights of edges between users and items in the user-item graph.
2. Retailrocket dataset recorded user interactions including clicks, adding to carts, and transactions with items on a real-world e-commerce website. We filtered out users interacting with less than 10 items and items having less than 30 users to alleviate the sparsity problem.

Evaluation protocols. We adopt the *leave-one-out* evaluation method which is widely used in works [He *et al.*, 2017; He *et al.*, 2018]. We compute average Hit Ratio (HR) [Deshpande and Karypis, 2004] and Normalized Discounted Cumulative Gain (NDCG) [He *et al.*, 2017] by

[1]https://grouplens.org/datasets/movielens/

[2]https://www.kaggle.com/retailrocket/ecommerce-dataset

	K=2				K=6				K=10			
	MovieLens		Retailrocket		MovieLens		Retailrocket		MovieLens		Retailrocket	
Methods	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
POP	16.12	13.54	13.32	11.41	32.77	20.59	27.44	17.40	41.68	23.35	36.81	20.32
BPR	24.74	21.89	42.59	38.21	49.28	31.87	58.38	44.25	64.06	36.25	65.68	47.44
FISM	25.99	22.81	43.79	39.05	51.33	33.65	59.40	46.47	65.53	37.95	66.71	48.80
SMF	24.93	21.88	43.59	38.44	49.57	31.85	59.43	45.83	65.66	36.91	67.11	48.24
sRMGCN	26.36	23.12	44.78	38.73	49.65	32.66	59.53	46.32	66.47	37.93	67.77	48.96
NAIS	26.95	23.73	47.75	39.44	52.58	34.30	61.27	48.27	66.06	38.53	68.23	50.68
PGCN-C	**29.71**	**25.14**	46.93	43.48	53.59	35.35	64.88	49.36	**67.99**	**39.86**	72.62	51.52
PGCN-A	29.30	24.96	**49.24**	**44.39**	**54.15**	**35.62**	**66.89**	**51.89**	67.79	39.57	**73.33**	**54.00**

Table 3: Recommendation performance (%) of compared methods.

ranking 100 items (one selected testing instance and 99 sampled negative instances) for each user with predicted preference scores.

Baselines. We compare our approaches, PGCN-C and PGCN-A, with following recommendation methods.

- **POP**. This method ranks items by popularity [Koren and Bell, 2015].
- **BPR**. It adopts a pair-wise ranking loss function for MF models [Rendle *et al.*, 2009].
- **FISM**. This method takes advantage of features of items that a user has interacted with to obtain better representations for users [Kabbur *et al.*, 2013].
- **SMF**. Item embeddings are learned with a song2vec model and are used to compute item similarity as constraints in MF models [Cheng *et al.*, 2017].
- **sRMGCNN**. It is a multi-graph convolutional neural network imposing smoothness priors on graphs [Monti *et al.*, 2017].
- **NAIS**. NAIS adopts attention mechanism to aggregate features of items to represent users' profile [He *et al.*, 2018].

Settings. We implemented our approaches using TensorFlow. To reduce the complexity, we only considered neighbors within two hops and selected at most 100 neighbors for each type. Only 2-item subsequences were utilized in graph G^U. We chose the LeakyReLU activation function and utilized two pooling and convolution layers. We adopted the Adam optimizer to minimize our loss function and set the learning rate to 0.001. We chose the $l2$ regularization and the weight of it was set to 10^{-4}. The batch size was 512. To prepare the training dataset, we sampled four negative instances for each positive one. FISM and NAIS took all items that a user had interacted with as neighbors of him/her. sRMGCNN had three LSTM layers. For NAIS, the smoothing exponent was $\beta = 0.5$.

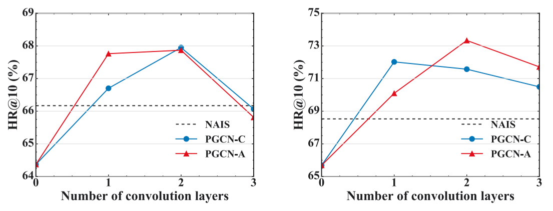
4.2 Experimental Results

Performance comparison (RQ1). Table 3 shows the performance of HR@K and NDCG@K with respect to the number of recommended items (i.e., K). For all methods, the size of latent vectors is set to 64. First, we can see that our approaches (PGCN-C and PGCN-A) achieve the best performance on the two datasets in terms of HR and NDCG. PGCN-A achieves a relative improvement of 4.7% and 6.8% over NAIS in terms of HR on MovieLens and Retailrocket, respectively. We also find that PGCN-A seems to have better performance on Retailrocket dataset than PGCN-C and the situation is contrary on MovieLens. This may be because the MovieLens dataset is small and PGCN-A which is more complex than PGCN-C may have overfitting problem on it. Among all baselines, NAIS utilizes attention mechanism to aggregate features of neighbors (items) to represent users' profile and performs the best. FISM also uses this kind of information. But it treats all items equally and its performance is a little poorer than NAIS. sRMGCN uses spectral graph convolution to retain the locality of user-item graph and performs better than other baselines.

Effect of heterogeneous graphs (RQ2). To demonstrate the effectiveness of the heterogeneous graphs, we compare results of our approaches leveraging one or two kinds of graphs. As shown in Table 4, PGCN-C can achieve the best performance with all the three kinds of graphs (i.e., G^R, G^I, and G^U) on both datasets. PGCN-A has the best performance using all graphs on Retailrocket dataset. The two approaches both have a better performance by adding graph G^I (indicating item dependency) or graph G^U (indicating user similarity) than approaches with only graph G^R on both datasets. Similar to Table 3, we can see that the performance improvement of adding G^I and G^U in Retailrocket is larger than that in MovieLens. This is because users may have many interactions with an item (buying one kind of item many times) and interaction sequences contain richer item dependency infor-

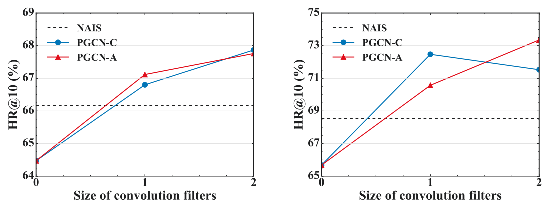
Graphs		G^R	G^R+G^I	G^R+G^U	G
MovieLens					
PGCN-C	HR	66.06	66.48	67.12	**67.97**
	NDCG	38.13	38.62	38.75	**39.42**
PGCN-A	HR	67.02	67.76	**68.39**	67.76
	NDCG	38.68	39.22	**39.75**	39.57
Retail					
PGCN-C	HR	69.00	71.35	69.60	**72.51**
	NDCG	49.36	50.58	49.96	**51.52**
PGCN-A	HR	68.83	70.58	69.80	**73.33**
	NDCG	50.15	52.21	52.02	**54.02**

Table 4: Performance (%) of PGCN with different graphs (K=10)



(a) MovieLens-HR (b) Retailrocket-HR

Figure 4: Performance of PGCN *w.r.t.* number of convolution layers.



(a) MovieLens-HR (b) Retail-HR

Figure 5: Performance of PGCN *w.r.t.* the size of convolution filters.

mation and user similarity on behaviors in Retailrocket.

Effect of the number of convolution layers (RQ3). As PGCN is a neural network based method, a problem worth studying is that how many neural layers are appropriate for the recommendation tasks. We evaluate our approaches by varying the number of neural layers (a neighbor pooling and a convolution layer are as one neural layer). Experimental results are shown in Figure 4. For the space limitation, we only show the performance of $HR@10$. When the number of convolution layer is 0, our approaches become traditional matrix factorization methods. From the figure, we can see that our approaches with convolution layers are much better than that without convolution layers on both datasets. As the number of layers grows, the performance grows. When the number of layers is 3, the performance drops. We think our approaches may have overfitting problem when the number is 3.

Effect of the filter size (RQ3). The size of convolution filters (i.e., L) can also affect the performance of our approaches. With larger convolution filters, our approaches can aggregate features of more neighbors. We show results of our approaches with different sizes of convolution filters in Figure 5. When the size is bigger, the number of neighbors grows exponentially. Therefore, we only evaluate the size of 1 and 2. Our approaches become traditional MF methods when the size of filters is 0. We can see that with convolution, our approaches achieve better performance on both datasets. The performance becomes better when the size of filters grows. For PGCN-C, the performance drops a little on Retailrocket dataset when the size of filters is 2. We think convolution operation may bring noise when the filter size is too large.

5 Related Work

Matrix Factorization (MF) as one of the most popular recommendation technics is used to model users' preference based on explicit ratings [Koren *et al.*, 2009] or implicit feedbacks [Hu *et al.*, 2008], and is enhanced with advanced pair-wise loss function [Rendle *et al.*, 2009] and interaction function [He *et al.*, 2017]. Based on interaction data, features of neighbors defined by the interactions are also used for improving the performance [Kabbur *et al.*, 2013; He *et al.*, 2018]. Besides the direct interaction, dependency among items in interaction sequences is also modeled by various approaches including Markov Chain [Feng *et al.*, 2015], tensor decomposition [Cheng *et al.*, 2013; Li *et al.*, 2017], and RNN [Song *et al.*, 2016; Yoo *et al.*, 2017].

Besides the above technics, convolutional neural network is another effective method to take advantages of features of neighbors which has been proven in various domains including computer vision and natural language processing [LeCun *et al.*, 2015; Hinton *et al.*, 2012]. To deal with a varied number of neighbors in arbitrary graphs, researchers proposed to design new convolutions in spectral domain [Bruna *et al.*, 2013; Henaff *et al.*, 2015] or spatial domain [Bruna *et al.*, 2013]. Compared to spectral methods, spatial convolution approaches are more flexible and design spatial convolution kernels by sharing weights based on the distance between two vertices [Atwood and Towsley, 2016] or performing random walks on graphs [Hechtlinger *et al.*, 2017]. Nguyen [2018] and Simonovsky [2017] designed convolution filters whose weights were conditioned on edge types. The spectral and spatial convolutional networks have been both applied on the user-item interaction graph for recommendation [Monti *et al.*, 2017; Berg *et al.*, 2017]. But they only use direct interactions between users and items and omit item dependency and user similarity information.

6 Conclusion

In this paper, we converted interaction sequences to three heterogeneous graphs and proposed to mine three kinds of information for item recommendation. We aggregated features of neighbors defined with paths in the graphs, and proposed a convolutional network, PGCN, which has novel neighbor pooling and convolution operations for learning shared representations of users and items in the heterogeneous graphs. The operations can retain the locality of graphs and deal with the varied number of neighbors. Experiments on two datasets showed that our approach outperformed other methods. Besides recommendation, PGCN is a general method and can be applied in other domains.

Acknowledgements

This research is supported in part by NSFC (No. 61772341, 61472254, 61772338) and STSCM (No. 18511103002). This work is also supported by the Program for Changjiang Young Scholars in University of China, the Program for China Top Young Talents, the Program for Shanghai Top Young Talents, Shanghai Engineering Research Center of Digital Education Equipment, and SJTU Global Strategic Partnership Fund (2019 SJTU-HKUST). Yanyan Shen is supported by NSFC No. 61602297.

References

[Atwood and Towsley, 2016] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[Berg *et al.*, 2017] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[Bruna *et al.*, 2013] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[Cheng *et al.*, 2013] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*, pages 2605–2611, 2013.

[Cheng *et al.*, 2017] Zhiyong Cheng, Jialie Shen, Lei Zhu, Mohan Kankanhalli, and Liqiang Nie. Exploiting music play sequence for music recommendation. In *IJCAI*, pages 3654–3660, 2017.

[Deshpande and Karypis, 2004] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[Feng *et al.*, 2015] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new poi recommendation. In *IJCAI*, pages 2069–2075, 2015.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[He *et al.*, 2018] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. Nais: Neural attentive item similarity model for recommendation. *TKDE*, 30(12):2354–2366, 2018.

[Hechtlinger *et al.*, 2017] Yotam Hechtlinger, Purvasha Chakravarti, and Jining Qin. A generalization of convolutional neural networks to graph-structured data. *stat*, 1050:26, 2017.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[Hinton *et al.*, 2012] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[Kabbur *et al.*, 2013] Santosh Kabbur, Xia Ning, and George Karypis. Fism: factored item similarity models for top-n recommender systems. In *KDD*, pages 659–667, 2013.

[Koren and Bell, 2015] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[Li *et al.*, 2017] Xin Li, Mingming Jiang, Huiting Hong, and Lejian Liao. A time-aware personalized point-of-interest recommendation via high-order tensor factorization. *TOIS*, 35(4):31, 2017.

[Monti *et al.*, 2017] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017.

[Nguyen and Grishman, 2018] Thien Huu Nguyen and Ralph Grishman. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*, pages 5900–5907, 2018.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Simonovsky and Komodakis, 2017] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, pages 3693–3702, 2017.

[Song *et al.*, 2016] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. Multi-rate deep learning for temporal recommendation. In *SIGIR*, pages 909–912, 2016.

[Velickovic *et al.*, 2017] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wang and Wang, 2014] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636. ACM, 2014.

[Yoo *et al.*, 2017] Jaeyoon Yoo, Heonseok Ha, Jihun Yi, Jongha Ryu, Chanju Kim, Jung-Woo Ha, Young-Han Kim, and Sungroh Yoon. Energy-based sequence gans for recommendation and their connection to imitation learning. *arXiv preprint arXiv:1706.09200*, 2017.