Multi-Graph Convolutional Neural Networks for Representation Learning in Recommendation

Jianing Sun and Yingxue Zhang
Montreal Research Center
Huawei Noah's Ark Lab
Montreal, QC, Canada
{jianing.sun, yingxue.zhang}@huawei.com

Abstract

Personalized recommendation plays an important role in many online services. Substantial research has been dedicated to learning vector representations of users and items with Graph Convolutional Networks (GCNs), however, we argue that existing methods do not make full use of the information that is available from user-item interaction data and the similarities between user pairs and item pairs. In this work, we develop a graph convolution-based recommendation framework, named Multi-Graph Convolution Collaborative Filtering (Multi-GCCF). Multi-GCCF not only expressively models the high-order information via a bipartite user-item interaction graph, but integrates the proximal information by building and processing user-user and item-item graphs. We conduct extensive experiments on four publicly accessible benchmarks, showing significant improvements relative to several state-of-the-art collaborative filtering and graph neural network-based recommendation models.

1 Introduction

Rapid and accurate prediction of users' preferences is the ultimate goal of today's recommender systems [8]. The core method behind recommender systems is collaborative filtering (CF) [9, 5]. The basic assumptions underpinning collaborative filtering are that similar users tend to like the same item and items with similar audiences tend to receive similar ratings from an individual.

One of the most successful methods for performing collaborative filtering is matrix factorization (MF) [5, 3, 4]. More recently, deep learning models have been introduced to boost the performance of traditional MF models. However, as observed in [11], deep learning-based recommendation models are not sufficient to yield optimal embeddings because they consider only user and item features. There is no explicit incorporation of user-item *interactions* when developing embeddings. A second limitation of the deep learning models is the reliance on the explicit feedback from users, which is usually relatively sparse. Recent works by Ying et al. [12] and Wang et al. [11] have demonstrated the effectiveness of processing the bipartite graph, reporting improvements over the SOTA models.

Despite their effectiveness, we perceive two important limitations. *First*, these models ignore the intrinsic difference between the two types of nodes in the bipartite graph (users and items). When aggregating information from neighboring nodes in the graph during the embedding construction procedure, the architectures in [12, 11] combine the information in the same way, using a function that has no dependence on the nature of the node. However, there is an important intrinsic difference between users and items in a real environment. *Second*, user-user and item-item relationships are also very important signals. Although two-hop neighborhoods in the bipartite graph capture these to some extent, it is reasonable to assume that we can improve the recommendation quality by constructing and learning from graphs that directly model user-user and item-item relationships.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

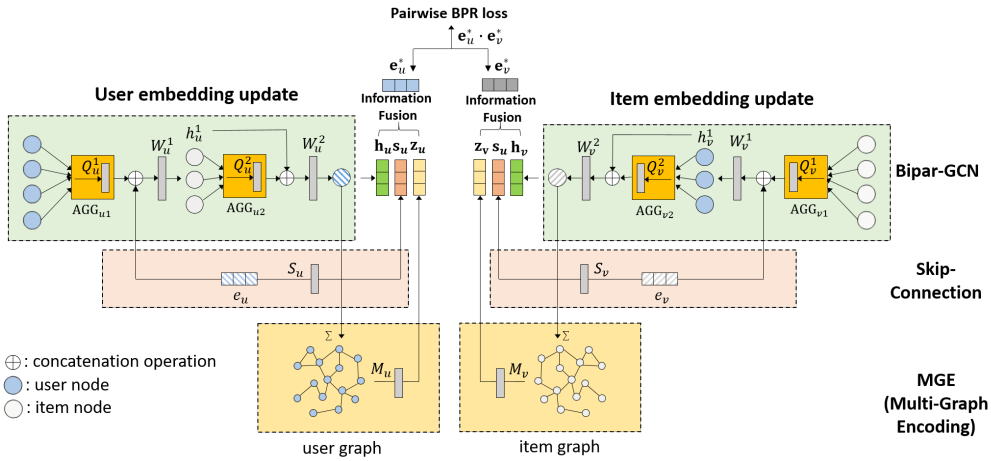


Figure 1: The overall architecture of Multi-GCCF.

In this paper, we propose a novel graph convolutional neural network (GCNN)-based recommender system framework, *Multi-GCCF*, with two key innovations: *1) Capturing the intrinsic difference between users and items*: we apply separate aggregation and transformation functions to process user nodes and item nodes when learning with a graph neural network. We find that the user and item embeddings are learned more precisely and the recommendation performance is improved. *2) Modeling user-user and item-item relationships explicitly*: we construct separate user-user and item-item graphs. Multi-GCCF conducts learning simultaneously on all three graphs and employs a multi-graph encoding layer to integrate the information provided by the user-item, user-user, and item-item graphs.

2 Methodology

In this section, we explain the three key components of our method. *First*, we develop a Bipartite Graph Convolutional Neural Network (Bipar-GCN) that acts as an encoder to generate user and item embeddings, by processing the user-item interaction (bipartite) graph. *Second*, a Multi-Graph Encoding layer (MGE) encodes latent information by constructing and processing multiple graphs. *Third*, a skip connection structure between the initial node feature and final embedding allows us to exploit any residual information in the raw feature that has not been captured by the graph processing. The overall framework of Multi-GCCF is depicted in Figure 1.

2.1 Bipartite Graph Convolutional Neural Networks

In a recommendation scenario, the user-item interaction can be readily formulated as a bipartite graph with two types of nodes. We apply a Bipartite Graph Convolutional Neural Network (Bipar-GCN) with one side representing user nodes and the other side representing item nodes. A figure illustrating Bipar-GCN is included in supplementary material. Taking a similar strategy as GraphSAGE [1], Bipar-GCN layer consists of two phases: *forward sampling* and *backward aggregating*.

After sampling the neighbors from layers 1 to K, Bipar-GCN encodes the user and item nodes by iteratively aggregating k-hop neighborhood information via graph convolution. There are initial embeddings $\mathbf{e}_u$ and $\mathbf{e}_v$ that are learned for each user u and item v. These embeddings are learned at the same time as the parameters of the GCNs. If there are informative input features $\mathbf{x}_u$ or $\mathbf{x}_v$, then the initial embedding can be a function of the features (e.g., the output of an MLP applied to $\mathbf{x}_u$). The layer-k embeddings of the target user u can be represented as:

$$\mathbf{h}_u^k = \sigma\Big(\mathbf{W}_u^k \cdot [\mathbf{h}_u^{k-1}; \mathbf{h}_{\mathcal{N}(u)}^{k-1}]\Big), \quad \mathbf{h}_u^0 = \mathbf{e}_u\,, \tag{1}$$

where $\mathbf{e}_u$ are the initial user embeddings, $[\ ;\ ]$ represents concatenation, $\sigma(\cdot)$ is the $\tanh$ activation function, $\mathbf{W}_u^k$ is the layer-k (user) transformation weight matrix shared across all user nodes. $\mathbf{h}_{\mathcal{N}(u)}^{k-1}$ is the learned neighborhood embedding. To achieve permutation invariance in the neighborhood, we

apply an element-wise weighted mean aggregator:

$$\mathbf{h}_{\mathcal{N}(u)}^{k-1} = \text{AGGREGATOR}_{\text{u}}\Big(\big\{\mathbf{h}_v^{k-1}, v \in \mathcal{N}(u)\big\} \Big), \tag{2}$$
$$\text{AGGREGATOR}_{\text{u}} = \sigma\left(\text{MEAN}\left(\big\{\mathbf{h}_v^{k-1} \cdot \mathbf{Q}_u^k, v \in \mathcal{N}(u)\big\}\right)\right).$$

Here $\mathbf{Q}_u^k$ is the layer-k (user) aggregator weight matrix, which is shared across all user nodes at layer k, and MEAN denotes the mean of the vectors in the argument set.

The vector representation of target item node v is generated similarly as user node but with another set of transformation and aggregator weight matrices.

2.2 Multi-Graph Encoding Layer

MGE layer generates an additional embedding for a target node by constructing two additional graphs and applying graph convolutional learning on them. This proximity information can make up for the very sparse user-item interaction bipartite graph. The user and item graphs are constructed by computing pairwise cosine similarities on the rows or columns of the rating matrix.

We generate embeddings for target nodes by aggregating the neighborhood features using a one-hop graph convolution layer and a sum aggregator:

$$\mathbf{z}_u = \sigma\Big(\sum_{i \in \mathcal{N}'(u)} \mathbf{e}_i \cdot \mathbf{M}_u \Big); \quad \mathbf{z}_v = \sigma\Big(\sum_{j \in \mathcal{N}'(v)} \mathbf{e}_j \cdot \mathbf{M}_v \Big). \tag{3}$$

Here $\mathcal{N}'(u)$ denotes the one-hop neighbourhood of user u in the user-user graph and $\mathcal{N}'(v)$ denotes the one-hop neighbourhood of item v in the item-item graph. $\mathbf{M}_u$ and $\mathbf{M}_v$ are the learnable user and item aggregation weight matrices, respectively.

When constructing the user and item similarity graph from the rating matrix, we select thresholds based on the cosine similarity that lead to an average degree of 10 for each graph. By merging the outputs of the Bipar-GCN and MGE layers together, we can take advantage of the different dependency relationships encoded by the three graphs. All three graphs can be easily constructed from historical interaction data alone, with very limited additional computation cost.

2.3 Skip-connection with Original Node Features

We further refine the embedding with information passed directly from the original node features. The intuition behind this is that both Bipar-GCN and MGE focus on extracting latent information based on relationships. As a result, the impact of the initial node features becomes less dominant. The skip connections allows the architecture to re-emphasize these features.

We pass the original features through a fully-connected layer to generate skip-connection embeddings:

$$\mathbf{s}_u = \sigma(\mathbf{e}_u \cdot \mathbf{S}_u); \qquad \mathbf{s}_v = \sigma(\mathbf{e}_v \cdot \mathbf{S}_v). \tag{4}$$

The bipartite-GCN, MGE layer and skip connections reveal latent information from three perspectives. It is important to determine how to merge these different embeddings effectively. We empirically find that element-wise sum performs much better than concatenation and attention (detailed ablation studies are included in supplementary material).

2.4 Model Training

We adapt our model to allow forward and backward propagation for mini-batches of triplet pairs $\{u, i, j\}$. To be more specific, we select unique user and item nodes u and $v = \{i, j\}$ from mini-batch pairs, then obtain low-dimensional embeddings $\{\mathbf{e}_u, \mathbf{e}_i, \mathbf{e}_j\}$ after information fusion, with stochastic gradient descent on the widely-used Bayesian Personalized Recommendation (BPR) [7] loss for optimizing recommendation models. The objective function is as follows:

$$loss = \sum_{(u,i,j) \in \mathcal{O}} -\log \sigma(\mathbf{e}_u^* \cdot \mathbf{e}_i^* - \mathbf{e}_u^* \cdot \mathbf{e}_j^*) + \lambda ||\Theta||_2^2 + \beta(||\mathbf{e}_u^*||_2^2 + ||\mathbf{e}_i^*||_2^2 + ||\mathbf{e}_j^*||_2^2) \tag{5}$$

where $\mathcal{O} = \{(u, i, j) | (u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-)\}$ denotes the training batch. $\mathcal{R}^+$ indicates observed positive interactions. $\mathcal{R}^-$ indicates sampled unobserved negative interactions. Θ is the model parameter set and $\mathbf{e}_u^*$, $\mathbf{e}_i^*$, and $\mathbf{e}_j^*$ are the learned embeddings. λ and β are regularization terms.

3 Experimental Evaluation

We perform experiments on four real-world datasets to evaluate our model. Further, we conduct extensive ablation studies on each proposed component (Bipar-GCN, MGE and skip connect), which is included in supplementary material. We also provide a visualization of the learned representation. Parameter settings is included in supplementary material.

3.1 Datasets and Evaluation Metrics

To evaluate the effectiveness of our method, we conduct extensive experiments on four benchmark datasets: *Gowalla*, *Amazon-Books, Amazon-CDs* and *Yelp2018*. These datasets are publicly accessible, real-world data with various domains, sizes, and sparsity. For all datasets, we filter out users and items with fewer than 10 interactions. Table 1 summarizes their statistics. For all experiments, we evaluate our model and baselines in terms of *Recall@*k and *NDCG@*k (we report *Recall@20* and *NDCG@20*). *Recall@*k indicates the coverage of true (preferred) items as a result of top-k recommendation. *NDCG@*k (normalized discounted cumulative gain) is a measure of ranking quality.

Table 1: Statistics of evaluation datasets.

Dataset	#User	#Items	#Interactions	Density
Gowalla	29,858	40,981	1,027,370	0.084%
Yelp2018	45,919	45,538	1,185,065	0.056%
Amazon-Books	52,643	91,599	2,984,108	0.062%
Amazon-CD	43,169	35,648	777,426	0.051%

3.2 Baseline Algorithms

We studied the performance of the following models: classical collaborative filtering methods(**BPRMF** [7] and **NeuMF** [2]); graph neural network-based collaborative filtering methods (**GC-MC**[10],**PinSage** [6] and **NGCF** [11]). Our proposed method, **Multi-GCCF**, contains two graph convolution layers on the user-item bipartite graph (2-hop aggregation), and one graph convolution layer on top of both the user-user graph and the item-item graph to model the similarities between user-pairs and item-pairs.

3.3 Comparison with Baselines

Table 2 reports the overall performance compared with baselines. Each result is the average performance from 5 runs with random weight initializations. We find that Multi-GCCF consistently yields the best performance for all datasets. More precisely, Multi-GCCF improves over the strongest baselines with respect to Recall@20 by **9.01%**, **12.19%**, **5.52%**, and **3.10%** for Yelp2018, Amazon-CDs, Amazon-Books and Gowalla, respectively. For the NDCG@20 metric, Multi-GCCF outperforms the next best method by around 20% for each dataset. This suggests that, exploiting the latent information by utilizing multiple graphs and efficiently integrating different embeddings, Multi-GCCF ranks relevant items higher in the recommendation list.

Table 2: The overall performance comparison. Underline indicates the second best model performance. Asterisks denote scenarios where a Wilcoxon signed rank test indicates a statistically significant difference between the scores of the best and second-best algorithms.

	Gowalla		Amazon-Books		Amazon-CDs		Yelp2018	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
BPRMF	0.1291	0.1878	0.0250	0.0518	0.0865	0.0849	0.0494	0.0662
NeuMF	0.1326	0.1985	0.0253	0.0535	0.0913	0.1043	0.0513	0.0719
GC-MC	0.1395	0.1960	0.0288	0.0551	0.1245	0.1158	0.0597	0.0741
PinSage	0.1380	0.1947	0.0283	0.0545	0.1236	0.1118	<u>0.0612</u>	<u>0.0795</u>
NGCF	<u>0.1547</u>	<u>0.2237</u>	<u>0.0344</u>	<u>0.0630</u>	<u>0.1239</u>	<u>0.1138</u>	0.0581	0.0719
Multi-GCCF (d=64)	***0.1595**	***0.2778**	***0.0363**	***0.0782**	***0.1390**	***0.1420**	***0.0667**	***0.0960**
% Fractional Improv.	3.10%	24.18%	5.52%	24.50%	12.19%	24.78%	9.01%	20.70%
Multi-GCCF (d=128)	***0.1649**	***0.2878**	***0.0391**	***0.0820**	***0.1543**	***0.1560**	***0.0688**	***0.1053**
% Fractional Improv.	6.59%	28.65%	13.66%	30.15%	24.54%	37.1%	12.41%	32.40%

4 Conclusion

In this paper we have presented a novel collaborative filtering procedure that incorporates multiple graphs to explicitly represent user-item, user-user and item-item relationships. The proposed model, Multi-GCCF, constructs three embeddings learned from different perspectives on the available data. Extensive experiments on four real-world datasets demonstrate the effectiveness of our approach.

References

[1] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proc. Adv. Neural Inf. Proc. Systems*, 2017.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua. Neural collaborative filtering. In *Proc. Int. Conf. World Wide Web*, 2017.

[3] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, 2004.

[4] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2008.

[5] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[6] Y. Qu, B. Fang, W. Zhang, R. Tang, M. Niu, H. Guo, Y. Yu, and X. He. Product-based neural networks for user response prediction over multi-field categorical data. *ACM Trans. Inf. Syst.*, 37(1):5:1–5:35, 2019.

[7] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. Conf. Uncertainty in Artificial Intelligence*, 2009.

[8] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[9] J. B. Schafer, D. Frankowski, J. L. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web, Methods and Strategies of Web Personalization*, 2007.

[10] R. van den Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2018.

[11] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua. Neural graph collaborative filtering. In *Proc. ACM Int. Conf. Research and Development in Information Retrieval*, 2019.

[12] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining*, 2018.

5 Supplementary Materials

5.1 Effect of Different Information Fusion Methods

As we obtain three embeddings from different perspectives, we compare different methods to summarize them into one vector: element-wise sum, concatenation, and attention. Table 3 shows the experimental results for Gowalla and Amazon-CDs. We make the following observations: Summation performs much better than concatenation and attention. Summation generates an embedding of the same dimension as the component embeddings and does not involve any additional learnable parameters. The additional flexibility of attention and concatenation may harm the generalization capability of the model.

Table 3: Comparison of different information fusion methods when $d = 128$.

	Gowalla		Amazon-CDs	
	Recall@20	NDCG@20	Recall@20	NDCG@20
element-wise sum	**0.1649**	**0.2878**	**0.1543**	**0.1560**
concatenation	0.1575	0.2748	0.1432	0.1509
attention	0.1615	0.2790	0.1426	0.1501

5.2 Ablation Analysis

To assess and verify the effectiveness of the individual components of our proposed Multi-GCCF model, we conduct an ablation analysis on Gowalla and Yelp2018 in Table 4. The table illustrates the performance contribution of each component. The output embedding size is 128 for all ablation experiments. We compare to $d = 64$ baselines because they outperform the $d = 128$ versions.

We make the following observations:

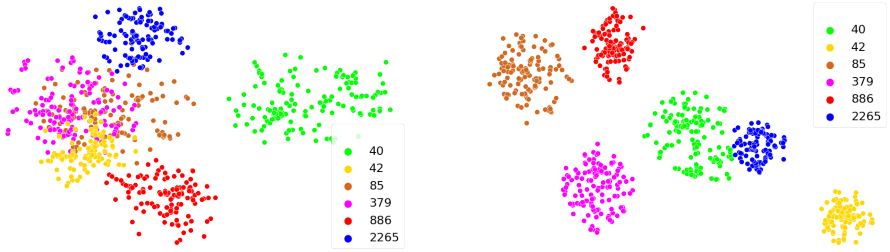
- All three main components of our proposed model, Bipar-GCN layer, MGE layer, and skip connection, are demonstrated to be effective.
- Our designed Bipar-GCN can greatly boost the performance with even one graph convolution layer on both the user side and the item side. Increasing the number of graph convolution layers can slightly improve the performance.
- Both MGE layer and skip connections lead to significant performance improvement.
- Combining all three components leads to further improvement, indicating that the different embeddings are effectively capturing different information about users, items, and user-item relationships.

Table 4: Ablation studies.

Architecture	Gowalla		Yelp2018	
	Recall@20	NDCG @20	Recall@20	NDCG@20
Best baseline (d=64)	0.1547	0.2237	0.0612	0.0795
Best baseline (d=128)	0.1435	0.2236	0.0527	0.0629
1-hop Bipar-GCN	0.1572	0.2673	0.0650	0.0903
2-hop Bipar-GCN	0.1582	0.2721	0.0661	0.0942
2-hop Bipar-GCN + skip connect	0.1603	0.2816	0.0675	0.0965
2-hop Bipar-GCN + MGE	0.1623	0.2820	0.0672	0.0982
Multi-GCCF (d=128)	**0.1649**	**0.2878**	**0.0688**	**0.1053**

5.3 Embedding Visualization

Figure 2 provides a visualization of the representations derived from BPRMF, NGCF and Multi-GCCF. Nodes with the same color represent all the item embeddings from one user's clicked/visited history, including test items that remain unobserved during training. We find that both BPRMF, NGCF and our proposed model have the tendency to encode the items that are preferred by the same user close to one another. However, Multi-GCCF generates tighter clusters, achieving a strong grouping effect for items that have been preferred by the same user.



(a) BPRMF (b) Multi-GCCF

Figure 2: Visualization of the t-SNE transformed representations derived from BPRMF and Multi-GCCF on Amazon-CDs. Numbers in the legend are user IDs. Points with the same color represent the relevant items from the corresponding user.

5.4 Figure illustrate of Bipar-GCN

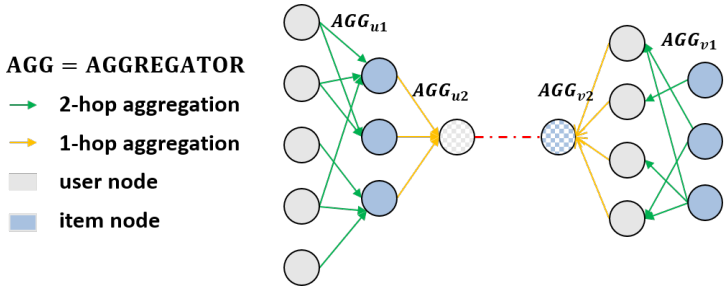


Figure 3: The accumulation of information in the bipartite user-item interaction graph. The circles with a mosaic pattern are target nodes (users on the left, items on the right) that are selected from the current training batch. Information is fused from the two-hop neighbours of a node. Only the embeddings of target nodes are updated during each iteration of the training procedure.

5.5 Parameter Settings

We optimize all models using the Adam optimizer with the Xavier initialization. The embedding size is fixed to 64 and the batch size to 1024, for all baseline models. Grid search is applied to choose the learning rate and the coefficient of L_2 normalization over the ranges $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and $\{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$, respectively. As in [11], for GC-MC and NGCF, we also tune the dropout rate and network structure. Pre-training [2] is used in NGCF and GC-MC to improve performance. We implement our Multi-GCCF model in PyTorch and use two Bipar-GCN layers with neighborhood sampling sizes $S_1 = 15$ and $S_2 = 10$. The output dimension of the first layer is fixed to 128; the final output dimension is selected from $\{64, 128\}$ for different experiments. We set the input node embedding dimension to 512. The neighborhood dropout ratio is set to 0.2. The regularization parameters in the objective function are set to $\lambda = 0.01$ and $\beta = 0.02$.