Collaborative Filtering on Bipartite Graphs using Graph Convolutional Networks

Minkyu Kim
Dept. of Big Data and Medical Convergence
Kangwon National University
Republic of Korea
minkyu.kim@kangwon.ac.kr

Jinho Kim
Dept. of Computer Science and Engineering
Kangwon National University
Republic of Korea
jhkim@kangwon.ac.kr

***Abstract*—Existing graph neural networks are not suitable to handle bipartite graphs, and existing graph-based collaborative filtering methods cannot model user-item interactions. To overcome these problems, we propose a novel collaborative filtering method named Graph Convolutional Collaborative Filtering (GCCF). Our GCCF aggregates neighbor nodes information with a novel Graph Convolutional Network (GCN) style node aggregation function improved for bipartite graphs. In addition, GCCF models user-item interactions using the user-item crossing layer. We conduct experiments to compare our GCCF with the existing state-of-the-art methods. Experimental results show that GCCF achieved or matched state-of-the-art results over five collaborative filtering datasets without any side information.**

***Index Terms*—Recommender Systems, Collaborative Filtering, Graph Neural Networks, Graph Convolutional Networks, Bipartite Graphs**

I. Introduction

Graph neural networks have succeeded in many research fields such as natural language processing, computer vision, and recommendation. Thanks to this success, many graph neural network-based collaborative filtering methods have been proposed recently. Collaborative filtering datasets can be represented as bipartite graphs. The adjacency matrix of a bipartite graph is asymmetric because the row and column of the adjacency matrix indicate different features (e.g., users and items). Therefore, despite node self-connections playing an important role in graph neural networks, we cannot represent self-connections in a bipartite graph. Due to this problem, Graph Convolutional Networks (GCNs) [1], and other graph neural networks are not suitable to handle bipartite graphs. Fig.1 shows node aggregation on a bipartite graph. Nodes of a bipartite graph can be divided into two disjoint sets. In addition, most existing graph-based collaborative filtering methods mainly focused on improving node aggregation function. They did not cover how to model user-item interactions. For example, Inductive Graph Matrix Completion (IGMC) [2] is a state-of-the-art collaborative filtering method. It aggregates neighbor nodes' information with the complicated function. However, the final output layer is a linear regressor. A linear regressor cannot model second-order interactions. Therefore, IGMC also cannot model user-item interactions.

To overcome these problems, we propose a novel graph-based collaborative filtering method named Graph Convolutional Collaborative Filtering (GCCF). Our GCCF aggregates neighbor nodes' information with a novel GCN style aggregation function improved for bipartite graphs. In addition, the final output layer of GCCF is the user-item crossing layer which can model user-item interactions. We conduct experiments to compare the proposed GCCF with the existing state-of-the-art methods. Experimental results show that GCCF achieved or matched state-of-the-art results over five collaborative filtering datasets without any side information (i.e., extra features). The source code of GCCF is publicly available[1].

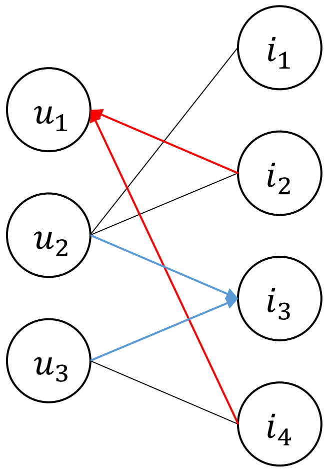

Fig. 1. Node Aggregation on a Bipartite Graph

The major contributions of this paper are as follows:

- We propose a novel graph-based collaborative filtering method named Graph Convolutional Collaborative Filtering (GCCF).
- Our GCCF aggregates neighbor nodes' information with a novel GCN style aggregation function improved for bipartite graphs and models user-item interactions using the user-item crossing layer.
- We demonstrate that GCCF achieved or matched state-of-the-art results over five datasets.

[1]https://github.com/gim4855744/GCCF

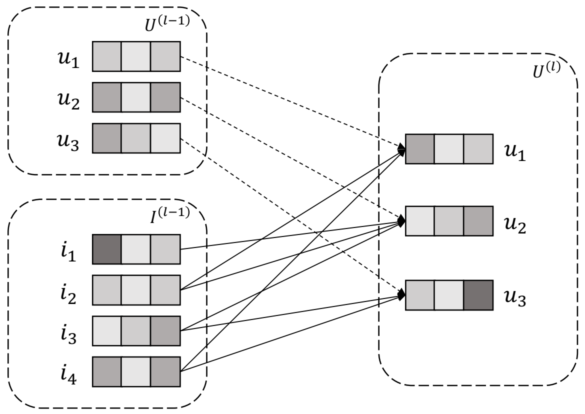

Fig. 2. Node Aggregation of a GCCF Layer

II. Related Works

A. Traditional Recommender Systems

Matrix Factorization (MF) [3]–[5] and Tensor Factorization (TF) [6], [7] are the most representative collaborative filtering methods. They decompose a matrix/tensor into several smaller matrices/tensors. The multiplication of the smaller matrices/tensors converges on the original matrix/tensor. SVD++ [8] is a matrix factorization method. It can consider users' and items' implicit feedbacks. timeSVD++ [9] is an extension of SVD++ to handle time series data. Factorization Machines (FMs) [10] are a latent factor method. FMs represent features as corresponding latent vectors.

B. Deep Learning based Recommender Systems

Recently, deep learning-based recommender systems are being actively studied. A collaborative filtering method using an auto-encoder has been proposed [11]. A deep learning-based matrix factorization method has been proposed [12]. To increase performance by using rich feature interactions, various methods combining several different methods have been proposed [13]–[16]. These combinatorial methods outperforms existing non-combinatorial deep learning-based methods.

C. Graph-based Recommender Systems

To handle graph-structured data, various graph-based recommender systems have been proposed. A graph-based method for social recommendations has been proposed [17]. Graph Convolutional Matrix Completion (GCMC) [18] is a matrix completion method using the graph auto-encoder. Inductive Graph Matrix Completion (IGMC) [2] is a graph-based matrix completion method. Various graph-based collaborative filtering methods have been proposed [19], [20].

III. Proposed Method

Many collaborative filtering datasets can be represented as bipartite graphs. The adjacency matrix of a bipartite graph is asymmetric because the row and column of the adjacency matrix indicate different features. Therefore, we cannot represent self-connections in the adjacency matrix. Due to this problem, Graph Convolutional Networks (GCNs) [1], and other graph neural networks are not suitable to handle bipartite graphs. In addition, existing graph-based collaborative filtering methods mainly focused on improving node aggregation function. They did not cover how to model user-item interactions. To overcome these problems, we propose a novel graph-based collaborative filtering method named Graph Convolutional Collaborative Filtering (GCCF). Our GCCF represents self-connections on a bipartite graph using residual connections and model user-item interactions using the user-item crossing layer.

First, we introduce the GCN-style node aggregation function on a bipartite graph. The aggregation function is defined as follows:

$$\mathbf{U}^{(l)} = \sigma(\hat{\mathbf{A}}_{user}\mathbf{I}^{(l-1)}\mathbf{W}^{(l)}_{user}) \tag{1}$$

$$\mathbf{I}^{(l)} = \sigma(\hat{\mathbf{A}}_{item}\mathbf{U}^{(l-1)}\mathbf{W}^{(l)}_{item}) \tag{2}$$

where $\hat{\mathbf{A}}_{user} \in \mathbb{R}^{M\times N}$ and $\hat{\mathbf{A}}_{item} \in \mathbb{R}^{N\times M}$ are the normalized adjacency matrices of users and items respectively, N and M are the numbers of users and items respectively, $\mathbf{U}^{(l)} \in \mathbb{R}^{M\times F}$ and $\mathbf{I}^{(l)} \in \mathbb{R}^{N\times F}$ are the embedding matrices of users and items in the l-th layer respectively, F is the embedding size, $\mathbf{W}^{(l)}_{user} \in \mathbb{R}^{F\times F}$ and $\mathbf{W}^{(l)}_{item} \in \mathbb{R}^{F\times F}$ are the trainable weight matrices of users and items in the l-th layer respectively, and $\sigma(\cdot)$ is an activation function. Nodes of a bipartite graph can be divided into two disjoint sets. Therefore, we need two parts for the aggregation function (i.e., the user and item aggregation parts). In addition, we use normalized adjacency matrices in the aggregation function. Our normalized method is defined as follows:

$$\hat{\mathbf{A}}_{user,i} = \frac{\mathbf{A}_{user,i}}{deg(\mathbf{A}_{user,i})} \tag{3}$$

$$\hat{\mathbf{A}}_{item,j} = \frac{\mathbf{A}_{item,j}}{deg(\mathbf{A}_{item,j})} \tag{4}$$

where $\mathbf{A}_{user}$ and $\mathbf{A}_{item}$ are the raw adjacency matrices of users and items respectively, and $deg(\mathbf{A}_{user,i})$ and $deg(\mathbf{A}_{item,j})$ are the degree of user i and item j respectively. Note that $\mathbf{A}_{user}$ is $\mathbf{A}^T_{item}$, but $\hat{\mathbf{A}}_{user}$ is not $\hat{\mathbf{A}}^T_{item}$. One problem of the node aggregation function defined above is that it cannot represent self-connections. To overcome this problem, we propose a novel GCN-style node aggregation function for bipartite graphs. The proposed function mimics self-connections using weighted residual connections. Our aggregation function (i.e. a GCCF layer) is defined as follows:

$$\mathbf{U}^{(l)} = \sigma(\hat{\mathbf{A}}_{user}\mathbf{I}^{(l-1)}\mathbf{W}^{(l)}_{user} + \mathbf{U}^{(l-1)}\mathbf{W}^{(l)}_{user}) \tag{5}$$

$$\mathbf{I}^{(l)} = \sigma(\hat{\mathbf{A}}_{item}\mathbf{U}^{(l-1)}\mathbf{W}^{(l)}_{item} + \mathbf{I}^{(l-1)}\mathbf{W}^{(l)}_{item}) \tag{6}$$

The embedding matrices $\mathbf{U}$ and $\mathbf{I}$ are self-connected to generate the next layer's embedding matrices. Fig.2 shows the architecture of a GCCF layer. Dotted arrows indicate the residual connections.

Finally, we introduce the user-item crossing layer to generate the final output value. Most graph-based collaborative filtering methods use a linear regressor as the final output

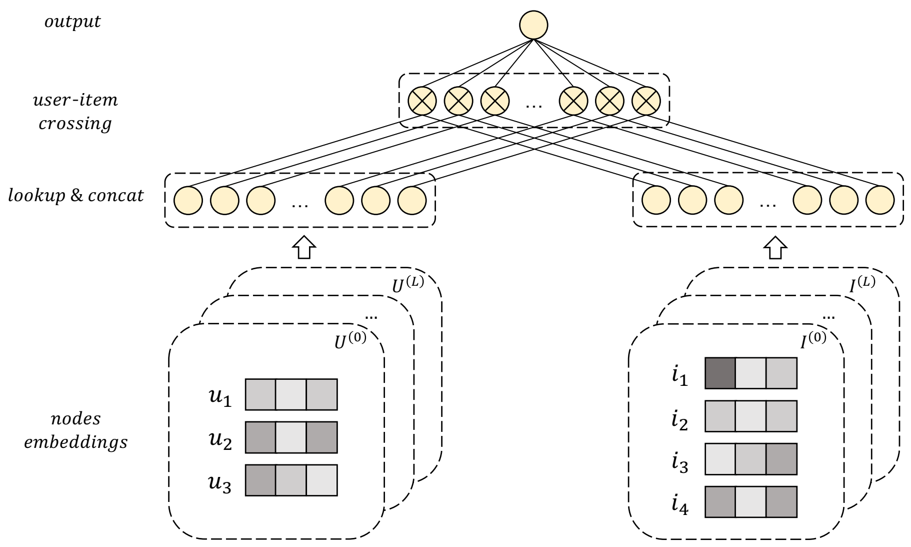

Fig. 3. The Architecture of Graph Convolutional Collaborative Filtering

layer. However, a linear regressor cannot model second-order interactions (e.g., user-item interactions). To overcome this problem, we propose the user-item crossing layer to model user-item interactions. The user-item crossing layer is defined as follows:

$$\mathbf{H} = \Big\Vert_{l=0}^{L} (\mathbf{U}_{\mathbf{u}}^{(l)} \odot \mathbf{I}_{\mathbf{i}}^{(l)}) \tag{7}$$

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{W}^{(out)} \tag{8}$$

where $\Vert$ is the concatenation operator, L is the number of layers, $\odot$ is the Hadamard product (i.e. element-wise product), **u** and **i** are the IDs of users and items respectively, and $\mathbf{W}^{(out)}$ is the trainable output matrix. The Hadamard product is a second-order operation. Therefore, we can model user-item interaction using this operation. In addition, we concatenate all layers for prediction. $\mathbf{W}^{(out)}$ distills useful information from modeled user-item interactions. Fig.3 shows the overall architecture of our GCCF. Algorithm 1 is the pseudocode of forward propagation of GCCF.

Algorithm 1 Forward Propagation of GCCF

```
1: % Normalizing Adjacency Matrices
2: $\hat{\mathbf{A}}_{user} \leftarrow \mathbf{A}_{user}/\text{sum}(\mathbf{A}_{user}, \text{axis=1})$
3: $\hat{\mathbf{A}}_{item} \leftarrow \mathbf{A}_{item}/\text{sum}(\mathbf{A}_{item}, \text{axis=1})$
4: % Node Aggregation
5: for $l \leftarrow 1$ to $L$ do
6:     $\mathbf{U}^{(l)} \leftarrow \sigma(\hat{\mathbf{A}}_{user}\mathbf{I}^{(l-1)}\mathbf{W}_{user}^{(l)} + \mathbf{U}^{(l-1)}\mathbf{W}_{user}^{(l)})$
7:     $\mathbf{I}^{(l)} \leftarrow \sigma(\hat{\mathbf{A}}_{item}\mathbf{U}^{(l-1)}\mathbf{W}_{item}^{(l)} + \mathbf{I}^{(l-1)}\mathbf{W}_{item}^{(l)})$
8: end for
9: % User-Item Interaction
10: $\mathbf{H} \leftarrow []$
11: for $l \leftarrow 0$ to $L$ do
12:     % embedding lookup and element-wise product
13:     $\mathbf{H}$.append($\mathbf{U}^{(l)}[\mathbf{u}] \odot \mathbf{I}^{(l)}[\mathbf{i}]$)
14: end for
15: % Prediction
16: $\mathbf{H} \leftarrow \text{concat}(\mathbf{H}, \text{axis=1})$
17: $\hat{\mathbf{y}} \leftarrow \mathbf{H}\mathbf{W}^{(out)}$
18: return $\hat{\mathbf{y}}$
```

IV. Experiments

A. Datasets

TABLE I
Statistics of Datasets

Dataset	# Ratings	# Users	# Items	Rating Type
ML-100K	100,000	943	1,682	1, 2, ..., 5
ML-1M	1,000,209	6,040	3,706	1, 2, ..., 5
Flixster	26,173	3,000	3,000	0.5, 1, ..., 5
Douban	136,891	3,000	3,000	1, 2, ..., 5
YahooMusic	5,335	3,000	3,000	1, 2, ..., 100

We evaluate various methods using five collaborative filtering datasets: ML-100k, ML-1M, Flixster, Douban, and YahooMusic. These datasets contain *(user, item, rating)* pairs and some side-information such as *age*, *sex*, or *genre*. We randomly split 90%/10% training/test sets for the ML-1M dataset. We use preprocessed training/test sets provided by [21] for other remaining datasets. TABLE I shows the statistical values of these experimental datasets in detail.

B. Experimental Setup

We implement our GCCF using PyTorch library. We use Adam optimizer to optimize the trainable parameters of a GCCF model. We train the GCCF model over 1000 epochs. Using the greedy search algorithm, we optimize the hyper-parameters of the GCCF model, such as learning rate, weight decay, and dropout rate. Detailed hyper-parameters settings are provided with the source code. The loss function is Mean Squared Error (MSE). Root Mean Squared Error (RMSE) is used for the evaluation metric.

C. Evaluation

TABLE II
PERFORMANCE COMPARISONS

Model	ML-100K	ML-1M	Flixster	Douban	YahooMusic
GCMC	**0.905**	**0.832**	0.917	0.734	20.5
IGMC	**0.905**	0.857	**0.872**	**0.721**	19.1
GCCF	0.911	0.846	0.938	0.729	**18.7**

We compare our GCCF with two state-of-the-art graph-based collaborative filtering methods, Graph Convolutional Matrix Completion (GCMC) [18] and Inductive Graph Matrix Completion (IGMC) [2]. TABLE II shows the RMSEs of GCMC, IGMC, and GCCF on five datasets. GCMC shows the highest performances in the ML-100K and ML-1M. IGMC shows the highest performances in the ML-100K, Flixster, and Douban. However, GCMC needs side information for high-performance, and IGMC needs the complicated sub-graph sampling method and node aggregation function. Our GCCF almost matched state-of-the-art results without any side information and additional sub-sampling method. In addition, GCCF achieved the new state-of-the-art result in the YahooMusic.

V. Conclusion

Existing graph neural networks are not suitable for bipartite graphs, and existing graph-based collaborative filtering methods can not model user-item interactions. Therefore, this paper proposed a novel graph-based collaborative filtering method named Graph Convolutional Collaborative Filtering (GCCF) to overcome these problems. Our GCCF aggregates neighbor nodes with a novel Graph Convolutional Network (GCN) [1] style aggregation function improved for bipartite graphs. In addition, the final output layer of GCCF is the user-item crossing layer. This layer can model second-order interactions like user-item interactions. We compared the proposed GCCF with two state-of-the-art methods on five collaborative filtering datasets. Experimental results demonstrated that our GCCF achieved or matched state-of-the-art results without any side information and additional sub-sampling method.

Ackowledgement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2021R1F1A1059255).

References

[1] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[2] M. Zhang and Y. Chen, "Inductive Matrix Completion Based on Graph Neural Network," in *International Conference on Learning Representations (ICLR)*, 2020.

[3] P. Paatero and U. Tapper, "Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[4] D. D. Lee and H. S. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[6] M. Welling and M. Weber, "Positive Tensor Factorization," *Pattern Recognition Letters*, vol. 22, no. 12, pp. 1255–1261, 2001.

[7] A. Shashua and T. Hazan, "Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 792–799.

[8] Y. Koren, "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[9] ——, "Collaborative Filtering with Temporal Dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[10] S. Rendle, "Factorization Machines," in *2010 IEEE International Conference on Data Mining*, 2010.

[11] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative Denoising Auto-Encoders for Top-N Recommender Systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016.

[12] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep Matrix Factorization Models for Recommender Systems," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017.

[13] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & Deep Learning for Recommender Systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 2016.

[14] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & Cross Network for Ad Click Predictions," in *Proceedings of the ADKDD'17*, 2017.

[15] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

[16] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[17] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph Neural Networks for Social Recommendation," in *The World Wide Web Conference*, 2019.

[18] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph Convolutional Matrix Completion," in *KDD'18 Deep Learning Day*, 2018.

[19] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural Graph Collaborative Filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019.

[20] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020.

[21] F. Monti, M. Bronstein, and X. Bresson, "Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks," in *Advances in Neural Information Processing Systems*, 2017.