Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 1, No. 1, 2022

Graph Convolution Network Recommender Rystem with Feature Embedding

Zhengtian Luo

School of Southwest Minzu University, Chengdu 610000, China.

Abstract: Recommender system has become one of the most important parts of information services on the Internet today the model based on graph neural network has been proved to be a very effective method in collaborative filtering recommender system [1,15]. However, the recommender system based on graph network usually only uses the interactive relationship between users and items, and their feature information is often not used effectively. In this work, we combine the use of feature information with graph convolutional network to design a new recommendation algorithm, Graph Convolution Network recommender system with Feature Embedding, GCNFE. Tests on the Movie lens and Last-FM datasets show that the new model performs better than previous models in most cases.

Keywords: Graph Neural Network; Recommender System; Collaborative Filtering.

1. Introduction

With the rapid development of the Internet era, information technology not only brings convenience to people, but also brings some troubles to people's life, especially the information overload caused by data explosion. As an information filtering system, recommender system can not only effectively solve the problem of information overload, but also has certain practical significance for promoting production and improving the quality of life. The recommender system realizes user group feature matching by learning large-scale group data, so as to help users effectively identify the content of interest from massive data [2]. These days, Web applications including social software, e-commerce, search engines, news portals, etc. almost every service that provides content to users is equipped with a recommender system.

The core task of recommender system is to predict whether the user will interact with the item, so as to provide the item more in line with the user's preference to the user. As such, collaborative filtering (CF), which focuses on exploiting the past user-item interactions to achieve the prediction, remains to be a fundamental task towards effective personalized recommendation. Such a recommender system generally finds users who are similar to the target user by learning the interaction records between users and items, and looks for items that the target user may be interested in from their interaction records.

Graph Neural Network (GNN) is an emerging framework that uses deep learning to directly learn graph structure data in recent years. Its excellent performance has attracted scholars' high attention and in-depth exploration. By formulating certain strategies on the nodes and edges of the graph, GNN transforms the graph structure data into a normative and standard representation, and inputs it into a variety of different neural networks for training. It has achieved excellent results in such tasks as node classification, edge information dissemination and graph clustering. In recent years, GNN has demonstrated its potential in the field of recommendation, graph neural networks have become the new state-of-the-art approach of recommender systems.

In this paper, we design a collaborative filtering recommendation model based on graph neural network, and integrate feature information into the user embedding, so as to obtain better results.

2. Related work

Generally speaking, collaborative filtering recommender systems have two ways to calculate the similarity between users and items. One is through the characteristics and labels of users or items themselves, and the other is through their interaction records.

Classical models using the first method include factorization machines (FM) [3], Neural FM(NFM)[4], logistic regression(LR)[5] and so on. LR converts the recommended problem into a dichotomous classification problem similar to Click-Through-Rate prediction (CTR prediction), converts various features into feature vectors, and inputs the logistic regression model to obtain a predicted CTR. Then it sorts the results according to the predicted CTR. On the basis of LR, FM model adds the crossover of second-order features into the model to obtain the hidden vectors of corresponding features by training each dimension feature, and the weight of features is obtained by inner product operation between hidden vectors. NFM uses neural networks to replace the second-order implicit vector crossover operation in FM, which makes NFM more expressive. Go without saying, these models are also often used in Click-Through-Rate prediction tasks.

The second method is the mainstream of collaborative filtering recommender system. For how to calculate similarity through interactive records, scholars have different methods, such as clustering [6], Markov decision processes (MDP) [7], matrix factorization [8] and so on. In the graph network recommender system, the mainstream method is still based on matrix factorization to calculate the similarity.

The matrix factorization algorithm generates an implicit vector for each user and each item, then calculates the similarity score between vectors of the user and the candidate items, and finally generates the recommendation list from high to low. The implicit vectors of users and items are obtained by decomposing user-item interaction matrix or rating matrix. As shown in Figure 1, we decompose a user

rating matrix into a users' embedding matrix and an items' embedding matrix of vector length 2. That's where the name matrix factorization comes from.

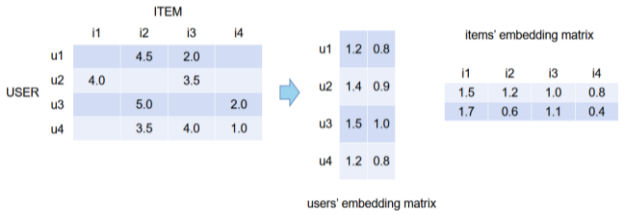

Figure 1. An illustration of matrix factorization

Because GNN contains the topological structure of the data, these structures can help the model to generate higher quality hidden vectors, so that the graph network recommender system (GNN-RecSYS) has a better effect. Graph network recommender system needs to obtain information from neighbor nodes to generate hidden vectors. When the degree of a node is very high, this operation will require a lot of computation, which makes it difficult for the model to obtain higher-order information. Graph-sage [9] solves this problem by randomly sampling the neighbor nodes of a node, and obtains the information of higher-order neighbors by random walk. NGCF[10] chooses to change the aggregation information of nodes. Each node transmits information to its neighbors. After several rounds of transmission, each node will have the information of its higher-order neighbors. The authors of LightGCN[11] believe that NGCF contains too many redundant operations inherited from graph convolution in the propagation of information, which can achieve better results when these operations are removed. Most of these models are trained using only historical interaction information, ignoring user and project characteristics. There are also scholars trying to use information from users and projects themselves. For example, GraphRec[12] introduces the user's social network graph, while KGAT[13] uses the feature information of some projects in the form of knowledge graph, but the effect is not particularly ideal.

3. Model

Considering that introducing user and project information into the model may improve the effect of the model, we propose the GCNFE model. Similar to the general graph network recommender system, our model is also divided into three parts: embedding layer, propagation layer and prediction layer. We mainly consider introducing user and project information in the embedding stage, and the approximate structure of the model is shown in Figure 2.

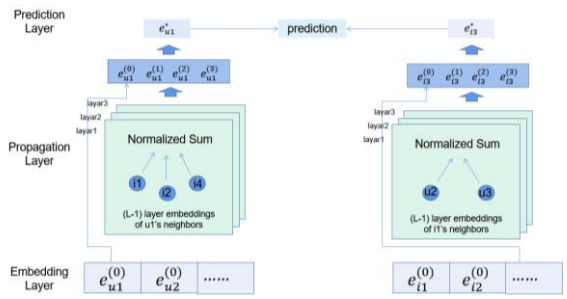

Figure 2. An illustration of GCNFE model architecture

3.1. Embedding layer

In the embedding layer, we generate a random vector for each user and project, and then connect the features of the user and project with their vectors after one-hot encoding. Since the vectors of users and items need to have the same length in the final prediction layer, a part of their vectors will be used to complement the length, as shown in equations (1) and (2):

$$e_{u1}^{(0)} = e_{random} || e_{u1's\ feature} || e_{padding1} \quad (1)$$

$$e_{i3}^{(0)} = e_{random} || e_{padding2} || e_{i3's\ feature} \quad (2)$$

In the equations, $e_{u1's\ feature}$ and $e_{i3's\ feature}$ are the vectors obtained by transforming the feature of the user and the item, and $e_{padding1}$, $e_{padding2}$ are used to complement the length of the embeddings. The embedding lengths of $e_{u1's\ feature}$ and $e_{padding2}$ are equal, and the embedding lengths of $e_{i3's\ feature}$ and $e_{padding1}$ are equal.

3.2. Propagation Layers

In the propagation layer, nodes will aggregate information about their neighbors and higher-order neighbors through message propagation. Let's take Figure 3 as an example.

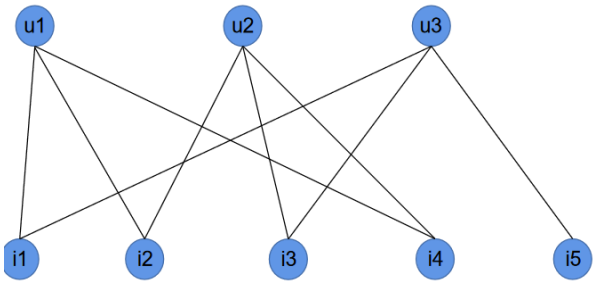

Figure 3. A subgraph the user-item interaction graph

Suppose we want to make a prediction for u1 and i3. In the first-layer propagation, u1 and i3 will aggregate the information of their first-order neighbors, as shown in equations (3) and (4) :

$$e_{u1}^{(1)} = \sum_{i \in N_{u1}} \frac{1}{\sqrt{|N_{u1}|}\sqrt{|N_i|}} e_i^{(0)} \quad (3)$$

$$e_{i3}^{(1)} = \sum_{u \in N_{i3}} \frac{1}{\sqrt{|N_{i3}|}\sqrt{|N_u|}} e_u^{(0)} \quad (4)$$

In the equations, $\frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}}$ is inherited from the symmetric normalization operation in the graph convolutional network, $|N_i|$ and $|N_u|$ represents the degree of the project and user node, respectively.

After a layer of propagation and aggregation, all nodes will get the information of their neighbor nodes. After another aggregation, they can get the information of their second-order neighbors from the neighbor nodes. After several aggregation, they can get the information of their higher-order neighbors. The equations for the k-layer aggregation is shown in (5) and (6) :

$$e_u^{(k)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^{(k-1)} \quad (5)$$

$$e_i^{(k)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(k-1)} \quad (6)$$

After K layer, a total of K+1 embeddings including the 0-th layer will be obtained. We will sum them and take the average as the final embedding. Again, using u1 and i3 as example, the results are shown in equations (7) and (8):

$$e_{u1}^{*} = \sum_{k=0}^{K} \frac{e_{u1}^{(k)}}{K+1} \quad (7)$$

$$e_{i3}^{*} = \sum_{k=0}^{K} \frac{e_{i3}^{(k)}}{K+1} \quad (8)$$

3.3. Model Prediction

In the prediction layer, we conduct the inner product operation between the vectors of users and items, and the result is used as the score of recommendation ranking.The results are shown in equations (9) :

$$y_{u1,i3} = {e_{u1}^{*}}^{T} e_{i3}^{*} \quad (9)$$

3.4. Model Training

During training, we choose Bayesian Personalized Ranking (BPR) loss[14] as our loss function, as shown in Equation (10) :

$$\text{Loss} = \sum_{(u,i,j)\in O} -\ln \sigma(y_{u,i} - y_{u,j}) + \lambda||\Theta||^2 \quad (10)$$

where O = {(u, i, j)|(u, i) ∈ R+, (u, j) ∈ R-} denotes the pairwise training data, R+ indicates the observed interactions, and R- is the unobserved interactions; σ(·) is the sigmoid function; Θ denotes all trainable model parameters, and λ controls the L2 regularization strength to prevent overfitting.

BPR loss has been intensively used in recommender systems. It takes into account the relative order between observed and unobserved user-item interactions. To be more specifically, BPR assumes that observed interactions (which are more reflective of user preferences) should be assigned higher predictive values than unobserved interactions.

4. Experiment and Results

We first introduce several data sets used in the experiment in Section 4.1, then introduce several methods of comparison with our model in Section 4.2, introduce the setting of some hyperparameters in the experiment in Section 4.3, and finally show the experimental comparison results.

4.1. Dataset

We used three datasets for our experiment:

Table1. Statistics of the datasets

Dataset	#Users	#Items	#Interactions	Sparsity
ml-100k	944	1,683	100,000	93.71%
ml-1m	6041	3,707	1,000,209	95.53%
Last-fm	993	107,296	18,498,005	82.64%

MoiveLens[19]: MoiveLens dataset is a very commonly used dataset in recommender system research. It was collected by researchers from the MoiveLens website and contains anonymous user ratings of movies, user attributes and information about movies. The MoiveLens-100K (ml-100k) dataset contains 100,000 scores, and the MoiveLens-1M (ml-1m) contains 1 million scores. Although the two data sets were collected and curated by the same organization, there is no duplication of data in them.

Last-fm[20]:The music recommendation dataset, published by Lastfm, contains four years' worth of listening records from about 1,000 anonymous users, as well as user, song and artist attributes. It's collected from Last.fm online music systems.

4.2. Compared Methods

We compare our proposed model with the following models:

BPRMF [14] optimizes the BPR loss to learn the latent representations for users and items with matrix factorization (MF) framework. One of the classic models in recommender system.

NeuMF [16] replaces the dot product in MF model with a multi-layer perceptron to learn the match function of users and items.

NGCF[10] uses methods in GCN to extract information from the bipartite graph of user items, trains the model with this information to generate vectors of user items, and then uses these embeddings to make predictions.

LightGCN[11] has a similar structure with NGCF, but some operations inherited from GCN are simplified, which not only improves the training speed of the model, but also achieves better results.

4.3. Experimental Settings

Evaluation Metrics

For each user in the test set, we treat all the items that the user has not interacted with as the negative items. Then each method outputs the user's preference scores over all the items, except the positive ones used in the training set. To evaluate the effectiveness of top-K recommendation and preference ranking, we adopt two widely-used evaluation protocols [10, 17]: recall@K and ndcg@K. By default, we set K = 10.

Other Settings

We implement the proposed model and all the baselines with RecBole [18], which is a unified opensource framework to develop and reproduce recommendation algorithms. In training, we randomly select 80% of interactions as training data and 10% of interactions as validation data. The remaining 10% interactions are used for performance comparison. We uniformly sample one negative item for each positive instance to form the training set. To ensure a fair comparison, we optimize all the methods with Adam optimizer and carefully search the hyper-parameters of all the baselines.

4.4. Result

Table 2. The comparison of overall performance among GCNFE and competing methods

	BPRMF		NeuMF		NGCF		LightGCN		GCNFE	
	recall	ndcg	recall	ndcg	recall	ndcg	recall	ndcg	recall	ndcg
ml-100k	0.2388	0.2865	0.2406	0.2764	0.2513	0.3012	0.1845	0.2125	0.2407	0.2866
ml-1m	0.1607	0.2543	0.1428	0.2324	0.1633	0.2542	0.1608	0.2582	0.1729	0.2681
Last-fm	0.0516	0.8686	0.0443	0.7939	0.0558	0.8798	0.0586	0.9044	0.0597	0.9049

Our experimental results are shown in Table 2. It can be seen that our model has the best effect in most cases, only slightly inferior to NGCF model on ml-100k dataset. This shows that we can improve the training effect of the model by introducing more information into the model in the vector embedding layer.

5. Conclusion

We propose a new graph network recommender system model, GCNFE. In the vector embedding stage, user information is encoded by one-hot and combined with randomly generated user embeddings and item embeddings, so that the model can obtain more information. We compared our model with the other four models on ml-100k, ml-1m and last-fm datasets, and proved the effectiveness of our model.

References

[1] Gao C , Zheng Y , Li N , et al. Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions[J]. 2021.

[2] RESNICK P, VARIAN H R. Recommender systems [J]. Commun ACM, 1997, 40(3): 56–58.

[3] S. Rendle. Factorization Machines[C]. 2010 IEEE International Conference on Data Mining, 2010.

[4] He X , Chua T S . Neural Factorization Machines for Sparse Predictive Analytics[C]// ACM. ACM, 2017:355-364.

[5] Richardson, Matthew, Dominowska, et al. Predicting clicks: Estimating the click-through rate for new ads[C]// the 16th international conference. ACM, 2007.

[6] Gao M, Cao F Y, Huang J Z. A Cross Cluster-Based Collaborative Filtering Method for Recommendation[C]. 2013 IEEE International Conference on Information and Automation, 2013: 447-452.

[7] Shani G, Brafman R I, Heckerman D. An MDP-Based Recommender System [J]. Journal of Machine Learning Research, 2005, 6: 1265-1295.

[8] Koren Y , Bell R , Volinsky C . Matrix Factorization Techniques for Recommender Systems[J]. Computer, 2009, 42(8):30-37.

[9] Hamilton W L , Ying R , Leskovec J . Inductive Representation Learning on Large Graphs[C]// 2017.

[10] Wang X , He X , Wang M , et al. Neural Graph Collaborative Filtering[J]. ACM, 2019.

[11] He X , Deng K , Wang X , et al. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation[C]// SIGIR '20: The 43rd International ACM SIGIR conference on research and development in Information Retrieval. ACM, 2020.

[12] Fan W , Yao M , Li Q , et al. Graph Neural Networks for Social Recommendation[J]. 2019.

[13] Wang X , He X , Cao Y , et al. KGAT: Knowledge Graph Attention Network for Recommendation[C]// Knowledge Discovery and Data Mining. ACM, 2019.

[14] S Rendle, C Freudenthaler, Z Gantner, et al. BPR: Bayesian personalized ranking from implicit feedback[C]// UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009. AUAI Press, 2009.

[15] Wu S , Sun F , Zhang W , et al. Graph Neural Networks in Recommender Systems: A Survey[J]. 2020.

[16] He X , Liao L , Zhang H , et al. Neural Collaborative Filtering[J]. International World Wide Web Conferences Steering Committee, 2017.

[17] Yang J H , Chen C M , Wang C J , et al. HOP-rec: high-order proximity for implicit recommendation[C]// the 12th ACM Conference. ACM, 2018.

[18] Zhao W X , Mu S , Hou Y , et al. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms[C]. Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021.

[19] F, MAXWELL, HARPER, et al. The MovieLens Datasets: History and Context[J]. Acm Transactions on Interactive Intelligent Systems, 2015.

[20] Henning V , Reichelt J . Mendeley - A Last.fm For Research?[C]// eScience, 2008. eScience '08. IEEE Fourth International Conference on. IEEE, 2009.