A Neural Collaborative Filtering Model with Interaction-based Neighborhood

Ting Bai[1,2], Ji-Rong Wen[1,2], Jun Zhang[1,2], Wayne Xin Zhao[1,2,*]
[1]School of Information, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
{baiting,zhangjun}@ruc.edu.cn,{jirong.wen,batmanfly}@gmail.com

ABSTRACT

Recently, deep neural networks have been widely applied to recommender systems. A representative work is to utilize deep learning for modeling complex user-item interactions. However, similar to traditional latent factor models by factorizing user-item interactions, they tend to be ineffective to capture localized information. Localized information, such as *neighborhood*, is important to recommender systems in complementing the user-item interaction data. Based on this consideration, we propose a novel Neighborhood-based Neural Collaborative Filtering model (NNCF). To the best of our knowledge, it is the first time that the neighborhood information is integrated into the neural collaborative filtering methods. Extensive experiments on three real-world datasets demonstrate the effectiveness of our model for the implicit recommendation task.

KEYWORDS

Recommender systems; Deep neural network; Neighborhood information

1 INTRODUCTION

Due to the explosive growth of information, recommender systems have become increasingly important in various online services. There are two mainstream approaches to recommender systems, namely memory-based and model-based approaches. As the most typical model-based recommendation approach, Matrix Factorization (MF) represents users and items in a shared latent space, and reconstructs the user-item interaction using their latent vectors. A major problem of MF is that it adopts a simple linear factorization, which may not be sufficient to model the complex user-item interactions. To overcome this limitation, some pioneering studies apply deep learning techniques to recommender systems, including neural rating prediction [7], auto-encoder based recommender [9] and neural collaborative filtering [3]. These works mainly utilize deep learning techniques as a powerful data model, in which complex user-item interactions and auxiliary information can be modeled. However, they also inherit a noteworthy weakness from previous latent factor models which directly factorize user-item interactions (*e.g.,* MF): it is poor at identifying strong associations among a small set of closely related items [4], especially when data is highly sparse. Based on this consideration, the aim of this paper is to retain the capacity of neural network models in learning an arbitrary user-item interaction function, and meanwhile enhance its ability to leverage localized information in complementing the interaction data. To fulfill this purpose, we have to address two challenging issues: (1) *what kind of localized information should be used* and (2) *how to model such localized information in a deep learning recommendation model*. For the first issue, previous methods typically utilize the neighborhood information, *e.g.,* the classic ItemKNN algorithms. The neural network model [3] simultaneously captures the two-way information of user-item interaction, while the neighbors derived from previous K Nearest Neighbors (KNN) algorithms are based on single-way relation, either user-user or item-item relation. Such neighborhood information may not be in the best form to fit into the interaction-based neural model. For solving the first issue, we propose to identify the two-way neighbors based on the *interaction network*. Unlike KNN algorithms, we characterize the neighborhood information based on user-item interactions. We construct the neighborhood by using a community-based algorithm, which selects direct or community neighbors in a localized group of both users and items. For solving the second issue, we set an integration component by incorporating both interaction and neighborhood

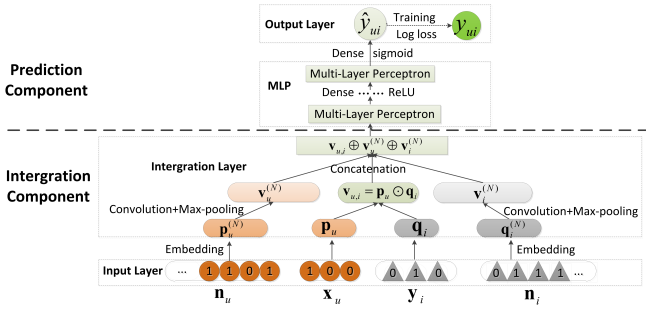


Figure 1: Overview of the architecture of NNCF.

* Corresponding Author.

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11...$15.00
https://doi.org/10.1145/3132847.3133083

information. By combining the two parts, our model jointly characterizes both interaction and neighborhood information in a unified neural network model.

Our contributions are summarized as follows: (1) We propose a model which can integrate neighborhood information into neural collaborative filtering for recommendation; (2) We propose to construct an interaction network which directly employs the interaction information to obtain the interaction-based neighborhood. (3) Extensive experiments on three real-world datasets demonstrate the effectiveness of our model for the implicit recommendation task.

2 PRELIMINARIES

Problem statement. Assume that we have a set of users and items, denoted by $\mathcal{U}$ and $\mathcal{I}$ respectively. Let $u \in \mathcal{U}$ denote a user and $i \in \mathcal{I}$ denote an item, and $y_{u,i}$ be the interaction label between u and i. Following [3], we use the general term of *interaction* to describe various kinds of relationship between users and items in different recommender systems, such as the purchase and play relationship on e-commerce and video-sharing websites. A value of 1 for $y_{u,i}$ indicates that u has interacted with i, and 0 otherwise. We call a user-item pair $\langle u, i\rangle$ an *interaction pair* when $y_{u,i} = 1$. Based on the above notations, the recommendation task can be defined as a prediction problem which aims to infer the value of the interaction label $y_{u,i}$ given an interaction pair $\langle u, i\rangle$. Usually, such a prediction problem can be reformulated as a ranking way: a true interaction pair (with the label of 1) should be ranked in a higher position than a false interaction pair (with the label of 0) in the recommendation list.

Construction of the interaction-based neighborhood. An interaction network is a bipartite graph formally defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V}$ consists of users and items vertices, *i.e.,* $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$, and the edge set $\mathcal{E}$ consists of the two-way interaction relations between users and items, and $(u, i) \in \mathcal{E}$ only if the interaction label $y_{u,i} = 1$. The interaction-based neighborhood for users is defined as a set of "associated" item vertices denoted by $\mathcal{I}^{(u)} \subset \mathcal{I}$. The interaction-based neighborhood for items is constructed by the same way.

A straightforward method to construct $\mathcal{U}^{(i)}$ or $\mathcal{I}^{(u)}$ is to collect all the linked vertices (*i.e., direct neighbors*) in the interaction network given a target vertex. However, such a method may not work well when the degree of a vertex is either too large or too small. We further propose a community-based algorithm to organize the vertices in the interaction network by communities. We use the Louvain method proposed in [8] to optimize the modularity of the interaction network with the partition parameter α. After community partition, each vertex will be assigned to a unique community. For efficiency consideration, only top K ranked neighbors will be selected. If the number of direct neighbors is less than K, we would add *community neighbors* to fill up.

3 NEIGHBORHOOD-BASED NEURAL COLLABORATIVE FILTERING MODEL

In this paper, we present a unified **N**eighborhood-based **N**eural **C**ollaborative **F**iltering model (**NNCF**). To give a global picture of our model, we present an illustrative figure for the proposed model in Fig. 1. We first introduce the integration component, which describes how to encode the user-item interaction pair together with the neighborhood information, and then introduce the prediction component based on Multi-Layer Perceptron (MLP).

3.1 Modeling Interaction and Neighborhood Information

As the input, our model takes in two kinds of information, namely user-item interaction and the neighborhood information.

Encoding the user-item pair. Formally, given a user-item pair $\langle u, i\rangle$, we encode the involved user u and item i using the one-hot representation, *i.e.,* $\mathbf{x}_u \in \mathbb{R}^{|\mathcal{U}|\times 1}$ and $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{I}|\times 1}$. In the one-hot vector $\mathbf{x}_u$ (or $\mathbf{y}_i$), only the u-th entry (or the i-th entry) is equal to 1. Similar to matrix factorization, two parameter matrices $\mathbf{P} \in \mathbb{R}^{K_1\times|\mathcal{U}|}$ and $\mathbf{Q} \in \mathbb{R}^{K_1\times|\mathcal{I}|}$ consist of the latent factors for users and items respectively. By applying a lookup layer, the one-hot user or item vector will be transformed into a latent vector as below

$$\mathbf{p}_u = \mathbf{P}^\top \cdot \mathbf{x}_u, \quad \mathbf{q}_i = \mathbf{Q}^\top \cdot \mathbf{y}_i, \tag{1}$$

To effectively capture the overall structure of user-item interaction, we follow the generalized MF method proposed in [3], and define the interaction function $\phi(\mathbf{p}_u, \mathbf{q}_i)$ as follows:

$$\mathbf{v}_{u,i} = \phi(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i, \tag{2}$$

where "$\odot$" denotes the element-wise product of vectors. The interaction function $\phi(\mathbf{p}_u, \mathbf{q}_i)$ applies a linear kernel to model the latent features interactions.

Encoding the neighborhood information. To encode the neighborhood information, we first code the neighbors using the one-hot representation. For a user u, given the neighborhood set $\mathcal{I}^{(u)}$ consisting of item vertices, we represent it using a $|\mathcal{I}|$-dimensional one-hot representation, denoted by $\mathbf{n}_u \in \mathbb{R}^{|\mathcal{I}|\times 1}$, only the entries corresponding to u's item neighbors will be set to 1. Similarly, the neighborhood set $\mathcal{U}^{(i)}$ of an item i is represented as one-hot vector $\mathbf{n}_i \in \mathbb{R}^{|\mathcal{U}|\times 1}$. Then we apply a concatenation-based lookup layer to transform both one-hot vectors into latent vectors

$$\mathbf{p}_u^{(N)} = \text{CONCAT-LOOKUP}(\mathbf{P'}^\top, \mathbf{n}_u), \tag{3}$$

$$\mathbf{q}_i^{(N)} = \text{CONCAT-LOOKUP}(\mathbf{Q'}^\top, \mathbf{n}_i), \tag{4}$$

where $\mathbf{P'} \in \mathbb{R}^{K_2\times|\mathcal{I}|}$ and $\mathbf{Q'} \in \mathbb{R}^{K_2\times|\mathcal{U}|}$ are the transformation matrices for lookup (similar to $\mathbf{P}$ and $\mathbf{Q}$). CONCAT-LOOKUP$(\cdot)$ is a function which first performs the lookup operation to obtain the corresponding embeddings for the input one-hot

representation vector, and then concatenates the multiple embeddings into a single vector as the output. The numbers of neighbors for different vertices are usually varying. To utilize such information, we have to transform the varying-length vector into a fixed-length one. We adopt the convolution operations on the latent vectors for neighbors, *i.e.,* $\mathbf{p}_u^{(N)}$ and $\mathbf{q}_i^{(N)}$. A convolution operation involves a filter which is applied to a fixed window to produce a new feature. Each possible window on the vector space produces a feature map. In order to retain the most important information, we further apply a max-pooling operation on the generated feature maps from convolution. Formally, we obtain the representations for the neighborhood information as follows

$$\mathbf{v}_u^{(N)} = \text{MP}(\text{CONV}(\mathbf{p}_u^{(N)})), \tag{5}$$

$$\mathbf{v}_i^{(N)} = \text{MP}(\text{CONV}(\mathbf{q}_i^{(N)})), \tag{6}$$

where $\mathbf{v}_u^{(N)} \in \mathbb{R}^{K_3 \times 1}$ and $\mathbf{v}_i^{(N)} \in \mathbb{R}^{K_3 \times 1}$ denote the latent vectors for the neighborhood of user u and item i respectively, and $\text{MP}(\cdot)$ and $\text{CONV}(\cdot)$. denote the max-pooling and convolution operations respectively.

Integrating interaction with neighborhood information. Once we have obtained the representations for the user-item pair and its neighborhood information, we further integrate these latent vectors into a unified representation as below

$$\widetilde{\mathbf{v}}_{u,i} = \mathbf{v}_{u,i} \oplus \mathbf{v}_u^{(N)} \oplus \mathbf{v}_i^{(N)}, \tag{7}$$

where "$\oplus$" denotes the vector concatenation operation, $\mathbf{v}_{u,i}$ (defined in Eq. 2) denotes the latent vector for the user-item pair $\langle u, i \rangle$, $\mathbf{v}_u^{(N)}$ (defined in Eq. 5) denotes the latent vector for the neighborhood of user u and $\mathbf{v}_i^{(N)}$ (defined in Eq. 6) denotes the latent vector for the neighborhood of item i.

3.2 MLP-based Prediction

The interaction between a user and an item can be very complex. Previous approaches usually assume a linear relation by decomposing the user-item matrix, *e.g.,* standard matrix factorization. We would like to endow our model a higher level of flexibility and nonlinearity to better characterize the interaction with the incorporation of neighborhood information. Hence, we propose to apply the MLP to model the user-item interactions. Generally, a MLP component can be constructed layer by layer. For $j = 1, ..., L$, we can have

$$\mathbf{z}_j = f^{(j)}(\mathbf{z}_{j-1}), \tag{8}$$

$$\hat{p}_{u,i} = \sigma(\mathbf{w}^\top \cdot \mathbf{z}_L), \tag{9}$$

where $f^{(j)}(\cdot)$ is the non-linear activation function for the j-th layer. We choose Rectifier Linear Unit (ReLU) as the activation function, which performs the best in our experiments. To feed the MLP, we set $\mathbf{z}_0 = \widetilde{\mathbf{v}}_{u,i}$ (defined in Eq. 7). For the output layer, $\mathbf{w}$ is the weights and $\sigma(\cdot)$ is the sigmoid function defined as $\sigma(x) = \frac{1}{1+\exp(-x)}$. $\hat{p}_{u,i}$ is the conditional probability of the interaction label being 1.

Table 1: Statistics of the evaluation datasets.

Datasets	#Interaction	# Users	#Items	Sparsity
Delicious	437,593	1,867	69,223	99.66%
MovieLens	1,000,209	3,706	6,040	95.53%
Rossmann	1,017,209	4,086	1,115	77.67%

The loss function for optimization. Given a training set $\mathcal{T} = \{< u, i, y_{u,i} >\}$, we adopt the cross-entropy loss as the optimization objective

$$\mathcal{L} = - \sum_{<u,i,y_{u,i}> \in \mathcal{T}} (y_{u,i} \cdot \log \hat{p}_{u,i} + (1 - y_{u,i}) \cdot \log(1 - \hat{p}_{u,i})), \tag{10}$$

where $\hat{p}_{u,i}$ is the conditional probability defined in Eq. 9. Note that our task is implicit recommendation, and we do not have explicit negative cases in training datasets. Following [3], we randomly sample four items that a user does not interact with as negative cases. To optimize the parameters for our model, we adopt the Stochastic Gradient Descent (SGD). The loss function in Eq. 10 is optimized by performing mini-batch Adam with a batch size of 1024, and the learning rate is set to 0.001. We implement our models in PYTHON using the library KERAS. We report the detailed parameter settings of our model below. The embedding size is set to 32 in the input layer, the number of kernels is set to 128 with a kernel size of 5 in the convolutional layer, and the embedding size is set to 128 in the last hidden layer. For all the nearest-neighbor methods, we select at most 50 neighbors for efficiency consideration, *i.e.,* $K = 50$.

4 EXPERIMENTS

Dataset. We experiment with three real-world datasets from different applications, namely Delicious [1], MovieLens [2], and Rossmann [6], which have been commonly adopted in recommendation tasks. The statistics of the three datasets are summarized in Table 1.

Evaluation metrics. We adopt the widely used leave-one-out method to perform the evaluation for recommendation [3, 4]. We randomly sample 100 negative items that a user has not interacted with and further combine the golden item (*i.e.,* the one that the user has interacted with) and the negative items into a randomly shuffled list. A comparison method will rank the list and return the top k ones as recommendations. Following [3], we adopt Hit ratio at rank k (HR@k) and Normalized Discounted Cumulative Gain at rank k (NDCG@k) to evaluate the performance of a ranked list.

Methods to compare. We consider the following baselines for performance comparisons. (1) *ItemPop*: It ranks the the items according to their popularity measured by the number of interactions [6]; (2) *ItemKNN*: It makes recommendations

Table 2: Performance comparisons of different methods on the recommendation task.

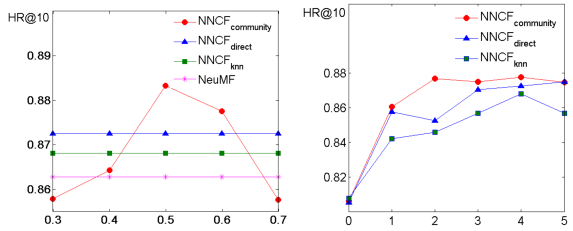
Datasets	Delicious				MovieLens				Rossmann			
Models	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10
ItemPop	0.0541	0.1044	0.0322	0.0483	0.3149	0.4503	0.2018	0.2524	0.0011	0.0027	0.0003	0.0008
ItemKNN	0.5969	0.6869	0.5590	0.6881	0.4501	0.6252	0.3014	0.3570	0.5836	0.6509	0.5150	0.5866
BPR	0.7377	0.7871	0.7411	0.8172	0.5103	0.6875	0.3621	0.4213	0.6175	0.6870	0.5640	0.6039
NeuMF	0.8553	0.8628	0.8068	0.8243	0.5655	0.7322	0.3830	0.4507	0.8496	0.9326	0.6566	0.6837
NNCF_{knn}	0.8478	0.8681	0.8305	0.8337	0.6059	0.7421	0.4121	0.4750	**0.8857**	0.9450	0.7438	0.7652
NNCF_{direct}	0.8597	0.8725	0.8323	0.8365	0.6146	0.7423	0.4160	0.4762	0.8757	0.9444	0.7390	0.7614
$\text{NNCF}_{community}$	**0.8731**	**0.8832**	**0.8458**	**0.8490**	**0.6200**	**0.7441**	**0.4221**	**0.4766**	0.8815	**0.9503**	**0.7465**	**0.7691**

according to the similarities of a candidate item to the past items [5]; (3) *BPR*: It optimizes the MF model with a pairwise ranking loss [6]; (4) *NeuMF*: It is the recently proposed neural network model for item recommendation [3].

It is noteworthy that our NNCF model itself is flexible to integrate neighborhood derived from other methods. Here, we consider three variants with different neighborhood construction algorithms, namely $\textbf{NNCF}_{knn}$, $\textbf{NNCF}_{direct}$, and $\textbf{NNCF}_{community}$ (See Section 2).

Results and analysis. We present the results of HR@k and NDCG@k on the recommendation performance in Table 2. We can have the following observations: (1) ItemPop is the weakest baseline, since it is a non-personalized method. ItemKNN and BPR perform better than ItemPop substantially. NeuMF is the best baseline, which represents the state-of-art performance on the recommendation task in the literature. (2) Our NNCF models perform better than all the baselines consistently on the three datasets, especially with the metrics of NDCG@k. This obversion demonstrates our proposed model is more capable of generating high-quality recommendations at very top positions. (3) By comparing the three NNCF variants, we can observe that the variant $\text{NNCF}_{community}$ gives the best performance in almost all the cases except HR@5 on the Rossmann dataset. The results indicate that the community-based algorithm is likely to generate better neighborhood for recommendation.

Effect of the number of communities and hidden layers. We examine the effect of different communities and hidden layers on the performance for our model. As shown in Fig. 2, we can see that (1) when $\alpha = 0.5$, NNCF achieves the best performance, which indicates that the number of communities should be set neither too small nor too large; and (2) NNCF achieves the best performance with four hidden layers, which is substantially better than that without hidden layers by incorporating nonlinear transformation.



(a) Varying α. (b) Varying the number of hidden layers.

Figure 2: Performance tuning with the varying of the community partition parameter α and numbers of hidden layers L.

5 CONCLUSION

This paper presented a novel neural network model NNCF, which jointly characterized both user-item interactions and neighborhood information for recommendation. To characterize neighborhood information, we proposed to use a community-based algorithm based on the interaction network. Extensive experiments on three real-world datasets demonstrated the effectiveness of our model with interaction-based neighbors for the implicit recommendation task, especially in the data sparsity situation.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61502502, the National Basic Research 973 Program of China under Grant No. 2014CB340403 and the Beijing Natural Science Foundation under Grant No. 4162032. Ting Bai was supported by the Outstanding Innovative Talents Cultivation Funded Programs 2016 of Renmin University of China.

REFERENCES

[1] Ivn Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. Second workshop on information heterogeneity and fusion in recommender systems. In *RecSys*. 387–389.
[2] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* 5, 4 (2016), 1–19.
[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[4] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.
[5] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.
[6] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.
[7] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*. 791–798.
[8] Vincent A. Traag. 2015. Faster unfolding of communities: speeding up the Louvain algorithm. In *CoRR*.
[9] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *KDD*. 1235–1244.