Outer Product-based Neural Collaborative Filtering

Xiangnan He[1], **Xiaoyu Du**[1,2], **Xiang Wang**[1], **Feng Tian**[3], **Jinhui Tang**[4] and **Tat-Seng Chua**[1]
[1] National University of Singapore
[2] Chengdu University of Information Technology
[3] Northeast Petroleum University
[4] Nanjing University of Science and Technology
{xiangnanhe, duxy.me}@gmail.com, xiangwang@u.nus.edu, dcscts@nus.edu.sg

Abstract

In this work, we contribute a new multi-layer neural network architecture named ONCF to perform collaborative filtering. The idea is to use an outer product to explicitly model the pairwise correlations between the dimensions of the embedding space. In contrast to existing neural recommender models that combine user embedding and item embedding via a simple concatenation or element-wise product, our proposal of using outer product above the embedding layer results in a two-dimensional *interaction map* that is more expressive and semantically plausible. Above the interaction map obtained by outer product, we propose to employ a convolutional neural network to learn high-order correlations among embedding dimensions. Extensive experiments on two public implicit feedback data demonstrate the effectiveness of our proposed ONCF framework, in particular, the positive effect of using outer product to model the correlations between embedding dimensions in the low level of multi-layer neural recommender model. [1]

1 Introduction

To facilitate the information seeking process for users in the age of data deluge, various information retrieval (IR) technologies have been widely deployed [Garcia-Molina *et al.*, 2011]. As a typical paradigm of information push, recommender systems have become a core service and a major monetization method for many customer-oriented systems [Wang *et al.*, 2018b]. Collaborative filtering (CF) is a key technique to build a personalized recommender system, which infers a user's preference not only from her behavior data but also the behavior data of other users. Among the various CF methods, model-based CF, more specifically, matrix factorization based methods [Rendle *et al.*, 2009; He *et al.*, 2016b; Zhang *et al.*, 2016] are known to provide superior performance over others and have become the mainstream of recommendation research.

[1]Work appeared in IJCAI 2018. The experiment codes are available at: https://github.com/duxy-me/ConvNCF

The key to design a CF model is in 1) how to represent a user and an item, and 2) how to model their interaction based on the representation. As a dominant model in CF, matrix factorization (MF) represents a user (or an item) as a vector of latent factors (also termed as *embedding*), and models an interaction as the inner product between the user embedding and item embedding. Many extensions have been developed for MF from both the modeling perspective [Wang *et al.*, 2015; Yu *et al.*, 2018; Wang *et al.*, 2018a] and learning perspective [Rendle *et al.*, 2009; Bayer *et al.*, 2017; He *et al.*, 2018]. For example, DeepMF [Xue *et al.*, 2017] extends MF by learning embeddings with deep neural networks, BPR [Rendle *et al.*, 2009] learns MF from implicit feedback with a pairwise ranking objective, and the recently proposed adversarial personalized ranking (APR) [He *et al.*, 2018] employs an adversarial training procedure to learn MF.

Despite its effectiveness and many subsequent developments, we point out that MF has an inherent limitation in its model design. Specifically, it uses a fixed and data-independent function — i.e., the inner product — as the interaction function [He *et al.*, 2017]. As a result, it essentially assumes that the embedding dimensions (i.e., dimensions of the embedding space) are independent with each other and contribute equally for the prediction of all data points. This assumption is impractical, since the embedding dimensions could be interpreted as certain properties of items [Zhang *et al.*, 2014], which are not necessarily to be independent. Moreover, this assumption has shown to be sub-optimal for learning from real-world feedback data that has rich yet complicated patterns, since several recent efforts on neural recommender models [Tay *et al.*, 2018; Bai *et al.*, 2017] have demonstrated that better recommendation performance can be obtained by learning the interaction function from data.

Among the neural network models for CF, neural matrix factorization (NeuMF) [He *et al.*, 2017] provides state-of-the-art performance by complementing the inner product with an adaptable multiple-layer perceptron (MLP) in learning the interaction function. Later on, using multiple nonlinear layers above the embedding layer has become a prevalent choice to learn the interaction function. Specifically, two common designs are placing a MLP above the concatenation [He *et al.*,

2017; Bai *et al.*, 2017] and the element-wise product [Zhang *et al.*, 2017; Wang *et al.*, 2017] of user embedding and item embedding. We argue that a potential limitation of such two designs is that there are few correlations between embedding dimensions being modeled. Although the following MLP is theoretically capable of learning any continuous function according to the universal approximation theorem [Hornik, 1991], there is no practical guarantee that the dimension correlations can be effectively captured with current optimization techniques.

In this work, we propose a new architecture for neural collaborative filtering (NCF) by integrating the correlations between embedding dimensions into modeling. Specifically, we propose to use an outer product operation above the embedding layer, explicitly capturing the pairwise correlations between embedding dimensions. We term the correlation matrix obtained by outer product as the *interaction map*, which is a $K \times K$ matrix where K denotes the embedding size. The interaction map is rather suitable for the CF task, since it not only subsumes the interaction signal used in MF (its diagonal elements correspond to the intermediate results of inner product), but also includes all other pairwise correlations. Such rich semantics in the interaction map facilitate the following non-linear layers to learn possible high-order dimension correlations. Moreover, the matrix form of the interaction map makes it feasible to learn the interaction function with the effective convolutional neural network (CNN), which is known to generalize better and is more easily to go deep than the fully connected MLP.

The contributions of this paper are as follows.

- We propose a new neural network framework ONCF, which supercharges NCF modeling with an outer product operation to model pairwise correlations between embedding dimensions.
- We propose a novel model named ConvNCF under the ONCF framework, which leverages CNN to learn high-order correlations among embedding dimensions from locally to globally in a hierarchical way.
- We conduct extensive experiments on two public implicit feedback data, which demonstrate the effectiveness and rationality of ONCF methods.
- This is the first work that uses CNN to learn the interaction function between user embedding and item embedding. It opens new doors of exploring the advanced and fastly evovling CNN methods for recommendation research.

2 Proposed Methods

We first present the **O**uter product based **N**eural **C**ollaborative **F**iltering (ONCF) framework. We then elaborate our proposed **Conv**olutional **NCF** (ConvNCF) model, an instantiation of ONCF that uses CNN to learn the interaction function based on the interaction map. Before delving into the technical details, we first introduce some basic notations.

Throughout the paper, we use bold uppercase letter (e.g., $\mathbf{P}$) to denote a matrix, bold lowercase letter to denote a vector (e.g., $\mathbf{p}_u$), and calligraphic uppercase letter to denote a tensor (e.g., $\mathcal{S}$). Moreover, scalar $p_{u,k}$ denotes the (u,k)-th element

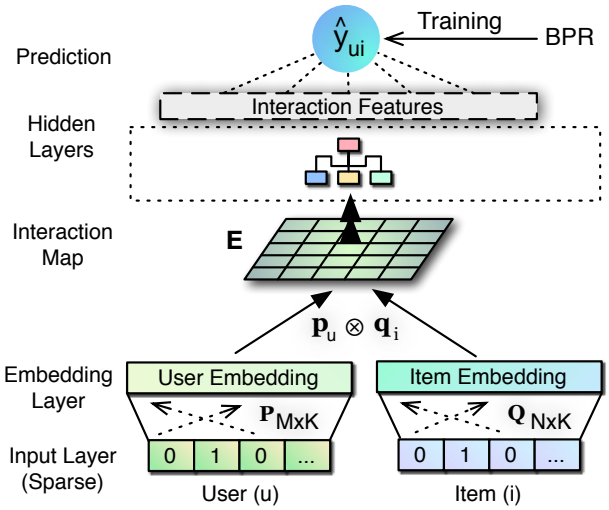

Figure 1: Outer Product-based NCF framework

of matrix $\mathbf{P}$, and vector $\mathbf{p}_u$ denotes the u-th row vector in $\mathbf{P}$. Let $\mathcal{S}$ be 3D tensor, then scalar $s_{a,b,c}$ denotes the (a,b,c)-th element of tensor $\mathcal{S}$, and vector $\mathbf{s}_{a,b}$ denotes the slice of $\mathcal{S}$ at the element (a,b).

2.1 ONCF framework

Figure 1 illustrates the ONCF framework. The target of modeling is to estimate the matching score between user u and item i, i.e., $\hat{y}_{ui}$; and then we can generate a personalized recommendation list of items for a user based on the scores.

Input and Embedding Layer. Given a user u and an item i and their features (e.g., ID, user gender, item category etc.), we first employ one-hot encoding on their features. Let $\mathbf{v}_u^U$ and $\mathbf{v}_i^I$ be the feature vector for user u and item i, respectively, we can obtain their embeddings $\mathbf{p}_u$ and $\mathbf{q}_i$ via

$$\mathbf{p}_u = \mathbf{P}^T \mathbf{v}_u^U, \quad \mathbf{q}_i = \mathbf{Q}^T \mathbf{v}_i^I, \tag{1}$$

where $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$ are the embedding matrix for user features and item features, respectively; K, M, and N denote the embedding size, number of user features, and number of item features, respectively. Note that in the pure CF case, only the ID feature will be used to describe a user and an item [He *et al.*, 2017], and thus M and N are the number of users and number of items, respectively.

Interaction Map. Above the embedding layer, we propose to use an outer product operation on $\mathbf{p}_u$ and $\mathbf{q}_i$ to obtain the interaction map:

$$\mathbf{E} = \mathbf{p}_u \otimes \mathbf{q}_i = \mathbf{p}_u \mathbf{q}_i^T, \tag{2}$$

where $\mathbf{E}$ is a $K \times K$ matrix, in which each element is evaluated as: $e_{k_1,k_2} = p_{u,k_1} q_{i,k_2}$.

This is the core design of our ONCF framework to ensure the effectiveness of ONCF for the recommendation task. Compared to existing recommender systems [He *et al.*, 2017; Zhang *et al.*, 2017], we argue that using outer product is more advantageous in threefold: 1) it subsumes matrix factorization (MF) — the dominant method for CF — which

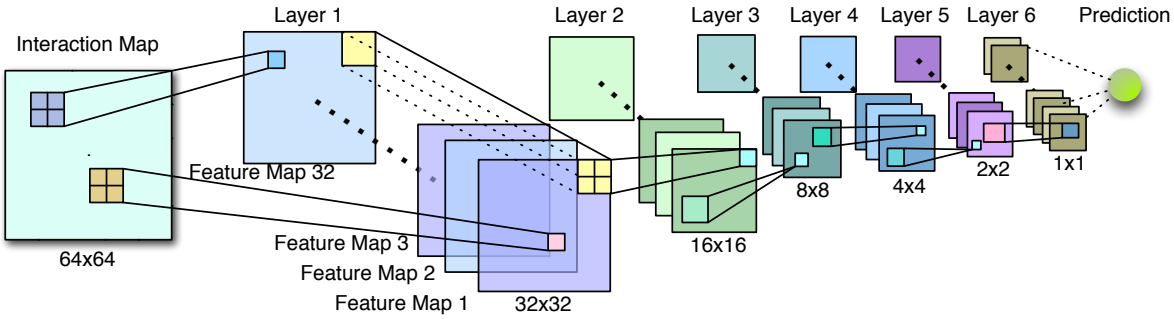

Figure 2: An example of the architecture of our ConvNCF model that has 6 convolution layers with embedding size 64.

considers only diagonal elements in our interaction map; 2) it encodes more signal than MF by accounting for the correlations between different embedding dimensions; and 3) it is more meaningful than the simple concatenation operation, which only retains the original information in embeddings without modeling any correlation. Moreover, it has been recently shown that, modeling the interaction of feature embeddings explicitly is particularly useful for a deep learning model to generalize well on sparse data, whereas using concatenation is sub-optimal [He and Chua, 2017; Beutel *et al.*, 2018].

Lastly, another potential benefit of the interaction map lies in its 2D matrix format — which is the same as an image. In this respect, the pairwise correlations encoded in the interaction map can be seen as the local features of an "image". As we all know, deep learning methods achieve the most success in computer vision domain, and many powerful deep models especially the ones based on CNN (e.g., ResNet [He *et al.*, 2016a] and DenseNet [Huang *et al.*, 2017]) have been developed for learning from 2D image data. Building a 2D interaction map allows these powerful CNN models to be also applied to learn the interaction function for the recommendation task.

Hidden Layers. Above the interaction map is a stack of hidden layers, which targets at extracting useful signal from the interaction map. It is subjected to design and can be abstracted as $\mathbf{g} = f_\Theta(\mathbf{E})$, where f_Θ denotes the model of hidden layers that has parameters Θ, and $\mathbf{g}$ is the output vector to be used for the final prediction. Technically speaking, f_Θ can be designed as any function that takes a matrix as input and outputs a vector. In Section 2.2, we elaborate how CNN can be employed to extract signal from the interaction map.

Prediction Layer. The prediction layer takes in vector $\mathbf{g}$ and outputs the prediction score as: $\hat{y}_{ui} = \mathbf{w}^T\mathbf{g}$, where vector $\mathbf{w}$ re-weights the interaction signal in $\mathbf{g}$. To summarize, the model parameters of our ONCF framework are $\Delta = \{\mathbf{P}, \mathbf{Q}, \Theta, \mathbf{w}\}$.

Learning ONCF for Personalized Ranking

Recommendation is a personalized ranking task. To this end, we consider learning parameters of ONCF with a ranking-aware objective. In the NCF paper [He *et al.*, 2017], the authors advocate the use of a pointwise classification loss to learn models from implicit feedback. However, another more reasonable assumption is that observed interactions should be ranked higher than the unobserved ones. To implement this idea, [Rendle *et al.*, 2009] proposed a Bayesian Personalized Ranking (BPR) objective function as follows:

$$L(\Delta) = \sum_{(u,i,j)\in\mathcal{D}} -\ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda_\Delta||\Delta||^2, \qquad (3)$$

where λ_Δ are parameter specific regularization hyperparameters to prevent overfitting, and $\mathcal{D}$ denotes the set of training instances: $\mathcal{D} := \{(u,i,j)|i \in \mathcal{Y}_u^+ \wedge j \notin \mathcal{Y}_u^+\}$, where $\mathcal{Y}_u^+$ denotes the set of items that has been consumed by user u. By minimizing the BPR loss, we tailor the ONCF framework for correctly predicting the relative orders between interactions, rather than their absolute scores as optimized in pointwise loss [He *et al.*, 2017; 2016b]. This can be more beneficial for addressing the personalized ranking task.

It is worth pointing out that in our ONCF framework, the weight vector $\mathbf{w}$ can control the magnitude of the value of $\hat{y}_{ui}$ for all predictions. As a result, scaling up $\mathbf{w}$ can increase the margin $\hat{y}_{ui} - \hat{y}_{uj}$ for all training instances and thus decrease the training loss. To avoid such trivial solution in optimizing ONCF, it is crucial to enforce L_2 regularization or the max-norm constraint on $\mathbf{w}$. Moreover, we are aware of other pairwise objectives have also been widely used for personalized ranking, such as the L2 square loss [Wang *et al.*, 2017]. We leave this exploration for ONCF as future work, as our initial experiments show that optimizing ONCF with the BPR objective leads to good top-k recommendation performance.

2.2 Convolutional NCF

Motivation: Drawback of MLP. In ONCF, the choice of hidden layers has a large impact on its performance. A straightforward solution is to use the MLP network as proposed in NCF [He *et al.*, 2017]; note that to apply MLP on the 2D interaction matrix $\mathbf{E} \in \mathbb{R}^{K\times K}$, we can flat $\mathbf{E}$ to a vector of size K^2. Despite that MLP is theoretically guaranteed to have a strong representation ability [Hornik, 1991], its main drawback of having a large number of parameters can not be ignored. As an example, assuming we set the embedding size of a ONCF model as 64 (i.e., $K = 64$) and follow the common practice of the half-size tower structure. In this case, even a 1-layer MLP has $8,388,608$ (i.e., $4,096 \times 2,048$) parameters, not to mention the use of more layers. We argue that such a large number of parameters makes MLP prohibitive to be used in ONCF because of three reasons: 1) It requires powerful machines with large memories to store the model; and 2) It

needs a large number of training data to learn the model well; and 3) It needs to be carefully tuned on the regularization of each layer to ensure good generalization performance[2].

The ConvNCF Model. To address the drawback of MLP, we propose to employ CNN above the interaction map to extract signals. As CNN stacks layers in a locally connected manner, it utilizes much fewer parameters than MLP. This allows us to build deeper models than MLP easily, and benefits the learning of high-order correlations among embedding dimensions. Figure 2 shows an illustrative example of our ConvNCF model. Note that due to the complicated concepts behind CNN (e.g., stride, padding etc.), we are not ambitious to give a systematic formulation of our ConvNCF model here. Instead, without loss of generality, we explain ConvNCF of this specific setting, since it has empirically shown good performance in our experiments. Technically speaking, any structure of CNN and parameter setting can be employed in our ConvNCF model. First, in Figure 2, the size of input interaction map is 64×64, and the model has 6 hidden layers, where each hidden layer has 32 feature maps. A feature map c in hidden layer l is represented as a 2D matrix $\mathbf{E}^{lc}$; since we set the stride to 2, the size of $\mathbf{E}^{lc}$ is half of its previous layer $l-1$, e.g. $\mathbf{E}^{1c} \in \mathbb{R}^{32\times32}$ and $\mathbf{E}^{2c} \in \mathbb{R}^{16\times16}$. All feature maps of Layer l can be represented as a 3D tensor $\mathcal{E}^l$.

Given the input interaction map $\mathbf{E}$, we can first get the feature maps of Layer 1 as follows:

$$\begin{aligned} \mathcal{E}^1 &= [e^1_{i,j,c}]_{32\times32\times32}, \quad \text{where} \\ e^1_{i,j,c} &= \text{ReLU}(b_1 + \sum_{a=0}^{1}\sum_{b=0}^{1} e_{2i+a,2j+b} \cdot \underbrace{t^1_{1-a,1-b,c}}_{\text{convolution filter}}), \end{aligned} \tag{4}$$

where b_1 denotes the bias term for Layer 1, and $\mathcal{T}^1 = [t^1_{a,b,c}]_{2\times2\times32}$ is a 3D tensor denoting the convolution filter for generating feature maps of Layer 1. We use the rectifer unit as activation function, a common choice in CNN to build deep models. Following the similar convolution operation, we can get the feature maps for the following layers. The only difference is that from Layer 1 on, the input to the next layer $l+1$ becomes a 3D tensor $\mathcal{E}^l$:

$$\begin{aligned} \mathcal{E}^{l+1} &= [e^{l+1}_{i,j,c}]_{s\times s\times32}, \quad \text{where } 1 \le l \le 5,\ s = \frac{64}{2^{l+1}}, \\ e^{l+1}_{i,j,c} &= \text{ReLU}(b_{l+1} + \sum_{a=0}^{1}\sum_{b=0}^{1} \mathbf{e}^{l}_{2i+a,2j+b} \cdot \mathbf{t}^{l+1}_{1-a,1-b,c}), \end{aligned} \tag{5}$$

where b_{l+1} denotes the bias term for Layer $l+1$, and $\mathcal{T}^{l+1} = [t^{l+1}_{a,b,c,d}]_{2\times2\times32\times32}$ denote the 4D convolution filter for Layer $l+1$. The output of the last layer is a tensor of dimension $1 \times 1 \times 32$, which can be seen as a vector and is projected to the final prediction score with a weight vector $\mathbf{w}$.

Note that convolution filter can be seen as the "locally connected weight matrix" for a layer, since it is shared in generating all entries of the feature maps of the layer. This significantly reduces the number of parameters of a convolutional layer compared to that of a fully connected layer. Specifically, in contrast to the 1-layer MLP that has over 8 millions parameters, the above 6-layer CNN has only about 20 thousands parameters, which are several magnitudes smaller. This makes our ConvNCF more stable and generalizable than MLP.

Rationality of ConvNCF. Here we give some intuitions on how ConvNCF can capture high-order correlations among embedding dimensions. In the interaction map $\mathbf{E}$, each entry e_{ij} encodes the second-order correlation between the dimension i and j. Next, each hidden layer l captures the correlations of a 2×2 local area[3] of its previous layer $l-1$. As an example, the entry $e^1_{x,y,c}$ in Layer 1 is dependent on four elements $[e_{2x,2y}; e_{2x,2y+1}; e_{2x+1,2y}; e_{2x+1,2y+1}]$, which means that it captures the 4-order correlations among the embedding dimensions $[2x; 2x+1; 2y; 2y+1]$. Following the same reasoning process, each entry in hidden layer l can be seen as capturing the correlations in a local area of size 2^l in the interaction map $\mathbf{E}$. As such, an entry in the last hidden layer encodes the correlations among all dimensions. Through this way of stacking multiple convolutional layers, we allow ConvNCF to learn high-order correlations among embedding dimensions from locally to globally, based on the 2D interaction map.

Training Details

We optimize ConvNCF with the BPR objective with mini-batch Adagrad [Duchi *et al.*, 2011]. Specifically, in each epoch, we first shuffle all observed interactions, and then get a mini-batch in a sequential way; given the mini-batch of observed interactions, we then generate negative examples on the fly to get the training triplets. The negative examples are randomly sampled from a uniform distribution; while recent efforts show that a better negative sampler can further improve the performance [Ding *et al.*, 2018], we leave this exploration as future work. We pre-train the embedding layer with MF. After pre-training, considering that other parameters of ConvNCF are randomly initialized and the overall model is in a underfitting state, we train ConvNCF for 1 epoch first without any regularization. For the following epochs, we enforce regularization on ConvNCF, including L_2 regularization on the embedding layer, convolution layers, and the output layer, respectively. Note that the regularization coefficients (especially for the output layer) have a very large impact on model performance.

3 Experiments

To comprehensively evaluate our proposed method, we conduct experiments to answer the following research questions:

RQ1 Can our proposed ConvNCF outperform the state-of-the-art recommendation methods?

RQ2 Are the proposed outer product operation and the CNN layer helpful for learning from user-item interaction data and improving the recommendation performance?

RQ3 How do the key hyperparameter in CNN (i.e., number of feature maps) affect ConvNCF's performance?

[2]In fact, another empirical evidence is that most papers used MLP with at most 3 hidden layers, and the performance only improves slightly (or even degrades) with more layers [He *et al.*, 2017; Covington *et al.*, 2016; He and Chua, 2017]

[3]The size of the local area is determined by our setting of the filter size, which is subjected to change with different settings.

	Gowalla						Yelp						
	HR@k			NDCG@k			HR@k			NDCG@k			RI
	$k=5$	$k=10$	$k=20$	$k=5$	$k=10$	$k=20$	$k=5$	$k=10$	$k=20$	$k=5$	$k=10$	$k=20$	
ItemPop	0.2003	0.2785	0.3739	0.1099	0.1350	0.1591	0.0710	0.1147	0.1732	0.0365	0.0505	0.0652	+227.6%
MF-BPR	0.6284	0.7480	0.8422	0.4825	0.5214	0.5454	0.1752	0.2817	0.4203	0.1104	0.1447	0.1796	+9.5%
MLP	0.6359	0.7590	0.8535	0.4802	0.5202	0.5443	0.1766	0.2831	0.4203	0.1103	0.1446	0.1792	+9.2%
JRL	0.6685	0.7747	0.8561	0.5270	0.5615	0.5821	0.1858	0.2922	0.4343	0.1177	0.1519	0.1877	+3.9%
NeuMF	0.6744	0.7793	0.8602	0.5319	0.5660	0.5865	0.1881	0.2958	0.4385	0.1189	0.1536	0.1895	+3.0%
ConvNCF	**0.6914***	**0.7936***	**0.8695***	**0.5494***	**0.5826***	**0.6019***	**0.1978***	**0.3086***	**0.4430***	**0.1243***	**0.1600***	**0.1939***	-

Table 1: Top-k recommendation performance where $k \in \{5, 10, 20\}$. RI indicates the average improvement of ConvNCF over the baseline. * indicates that the improvements over all other methods are statistically significant for $p < 0.05$.

3.1 Experimental Settings

Data Descriptions. We conduct experiments on two publicly accessible datasets: Yelp[4] and Gowalla[5].

Yelp. This is the Yelp Challenge data for user ratings on businesses. We filter the dataset following by [He *et al.*, 2016b]. Moreover, we merge the repetitive ratings at different timestamps to the earliest one, so as to study the performance of recommending novel items to a user. The final dataset obtains 25,815 users, 25,677 items, and 730,791 ratings.

Gowalla. This is the check-in dataset from Gowalla, a location-based social network, constructed by [Liang *et al.*, 2016] for item recommendation. To ensure the quality of the dataset, we perform a modest filtering on the data, retaining users with at least two interactions and items with at least ten interactions. The final dataset contains 54,156 users, 52,400 items, and 1,249,703 interactions.

Evaluation Protocols. For each user in the dataset, we holdout the latest one interaction as the testing positive sample, and then pair it with 999 items that the user did not rate before as the negative samples. Each method then generates predictions for these $1,000$ user-item interactions. To evaluate the results, we adopt two metrics *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG), same as [He *et al.*, 2017]. HR@k is a recall-based metric, measuring whether the testing item is in the top-k position (1 for yes and 0 otherwise). NDCG@k assigns the higher scores to the items within the top k positions of the ranking list. To eliminate the effect of random oscillation, we report the average scores of the last ten epochs after convergence.

Baselines. To justify the effectiveness of our proposed ConvNCF, we study the performance of the following methods:

1. ItemPop ranks the items based on their popularity, which is calculated by the number of interactions. It is always taken as a benchmark for recommender algorithms.

2. MF-BPR [Rendle *et al.*, 2009] optimizes the standard MF model with the pairwise BPR ranking loss.

3. MLP [He *et al.*, 2017] is a NCF method that concatenates user embedding and item embedding to feed to the standard MLP for learning the interaction function.

4. JRL [Zhang *et al.*, 2017] is a NCF method that places a MLP above the element-wise product of user embedding and item embedding. Its difference with GMF [He *et al.*, 2017] is that JRL uses multiple hidden layers above the element-wise product, while GMF directly outputs the prediction score.

5. NeuMF [He *et al.*, 2017] is the state-of-the-art method for item recommendation, which combines hidden layer of GMF and MLP to learn the user-item interaction function.

Parameter Settings. We implement our methods with Tensorflow, which is available at: https://github.com/duxyme/ConvNCF. We randomly holdout 1 training interaction for each user as the validation set to tune hyperparameters. We evaluate ConvNCF of the specific setting as illustrated in Figure 2. The regularization coefficients are separately tuned for the embedding layer, convolution layers, and output layer in the range of $[10^{-3}, 10^{-2}, ..., 10^{2}]$. For a fair comparison, we set the embedding size as 64 for all models and optimize them with the same BPR loss using mini-batch Adagrad (the learning rate is 0.05). For MLP, JRL and NeuMF that have multiple fully connected layers, we tuned the number of layers from 1 to 3 following the tower structure of [He *et al.*, 2017]. For all models besides MF-BPR, we pre-train their embedding layers using the MF-BPR, and the L_2 regularization for each method has been fairly tuned.

3.2 Performance Comparison (RQ1)

Table 1 shows the Top-k recommendation performance on both datasets where k is set to 5, 10, and 20. We have the following key observations:

- ConvNCF achieves the best performance in general, and obtains high improvements over the state-of-the-art methods. This justifies the utility of ONCF framework that uses outer product to obtain the 2D interaction map, and the efficacy of CNN in learning high-order correlations among embedding dimensions.
- JRL consistently outperforms MLP by a large margin on both datasets. This indicates that, explicitly modeling the correlations of embedding dimensions is rather helpful for the learning of the following hidden layers, even for simple correlations that assume dimensions are independent of each other. Meanwhile, it reveals the practical difficulties to train MLP well, although it has strong representation ability in principle [Hornik, 1991].

3.3 Efficacy of Outer Product and CNN (RQ2)

Due to space limitation, for the blow two studies, we only show the results of NDCG, and the results of HR admit the same trend thus they are omitted.

[4]https://github.com/hexiangnan/sigir16-eals

[5]http://dawenl.github.io/data/gowalla_pro.zip

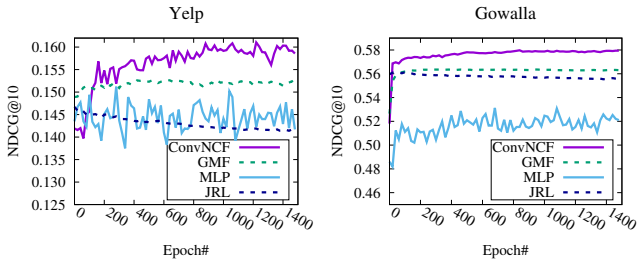

Figure 3: NDCG@10 of applying different operations above the embedding layer in each epoch (GMF and JRL use element-wise product, MLP uses concatenation, and ConvNCF uses outer product).

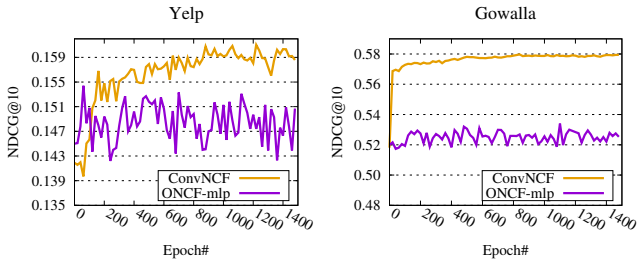

Figure 4: NDCG@10 of using different hidden layers for ONCF (ConvNCF uses a 6-layer CNN and ONCF-mlp uses a 3-layer MLP above the interaction map).

Efficacy of Outer Product. To show the effect of outer product, we replace it with the two common choices in existing solutions — concatenation (i.e., MLP) and element-wise product (i.e., GMF and JRL). We compare their performance with ConvNCF in each epoch in Figure 3. We observe that ConvNCF outperforms other methods by a large margin on both datasets, verifying the positive effect of using outer product above the embedding layer. Specifically, the improvements over GMF and JRL demonstrate that explicitly modeling the correlations between different embedding dimensions are useful. Lastly, the rather weak and unstable performance of MLP imply the difficulties to train MLP well, especially when the low-level has fewer semantics about the feature interactions. This is consistent with the recent finding of [He and Chua, 2017] in using MLP for sparse data prediction. .

Efficacy of CNN. To make a fair comparison between CNN and MLP under our ONCF framework, we use MLP to learn from the same interaction map generated by outer product. Specifically, we first flatten the interaction as a K^2 dimensional vector, and then place a 3-layer MLP above it. We term this method as ONCF-mlp. Figure 4 compares its performance with ConvNCF in each epoch. We can see that ONCF-mlp performs much worse than ConvNCF, in spite of the fact that it uses much more parameters (3 magnitudes) than ConvNCF. Another drawback of using such many parameters in ONCF-mlp is that it makes the model rather unstable, which is evidenced by its large variance in epoch. In contrast, our ConvNCF achieves much better and stable performance by using the locally connected CNN. These empirical evidence provide support for our motivation of designing ConvNCF and our discussion of MLP's drawbacks in Section 2.2.

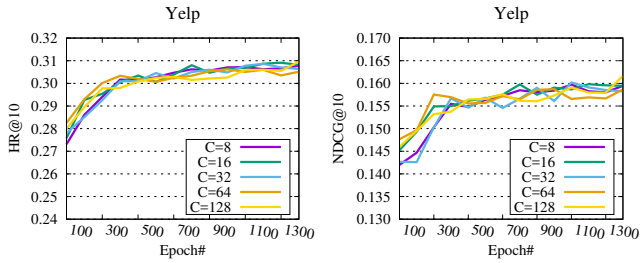

Figure 5: Performance of ConvNCF w.r.t. different numbers of feature maps per convolutional layer (denoted by C) in each epoch on Yelp.

3.4 Hyperparameter Study (RQ3)

Impact of Feature Map Number. The number of feature maps in each CNN layer affects the representation ability of our ConvNCF. Figure 5 shows the performance of ConvNCF with respect to different numbers of feature maps. We can see that all the curves increase steadily and finally achieve similar performance, though there are some slight differences on the convergence curve. This reflects the strong expressiveness and generalization of using CNN under the ONCF framework since dramatically increasing the number of parameters of a neural network does not lead to overfitting. Consequently, our model is very suitable for practical use.

4 Conclusion

We presented a new neural network framework for collaborative filtering, named ONCF. The special design of ONCF is the use of an outer product operation above the embedding layer, which results in a semantic-rich interaction map that encodes pairwise correlations between embedding dimensions. This facilitates the following deep layers learning high-order correlations among embedding dimensions. To demonstrate this utility, we proposed a new model under the ONCF framework, named ConvNCF, which uses multiple convolution layers above the interaction map. Extensive experiments on two real-world datasets show that ConvNCF outperforms state-of-the-art methods in top-k recommendation. In future, we will explore more advanced CNN models such as ResNet [He *et al.*, 2016a] and DenseNet [Huang *et al.*, 2017] to further explore the potentials of our ONCF framework. Moreover, we will extend ONCF to content-based recommendation scenarios [Chen *et al.*, 2017; Yu *et al.*, 2018], where the item features have richer semantics than just an ID. Particularly, we are interested in building recommender systems for multimedia items like images and videos, and textual items like news.

5 Acknowledgments

This work is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative, by the 973 Program of China under Project No.: 2014CB347600, by the Natural Science Foundation of China under Grant No.: 61732007, 61702300, 61501063, 61502094, and 61501064, by the Scientific Research Foundation of Science and Technology Department of Sichuan

Province under Grant No. 2016JY0240, and by the Natural Science Foundation of Heilongjiang Province of China (No.F2016002). Jinhui Tang is the corresponding author.

References

[Bai *et al.*, 2017] Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. A neural collaborative filtering model with interaction-based neighborhood. In *CIKM*, pages 1979–1982, 2017.

[Bayer *et al.*, 2017] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, pages 1341–1350, 2017.

[Beutel *et al.*, 2018] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*, pages 46–54, 2018.

[Chen *et al.*, 2017] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *SIGIR*, pages 335–344, 2017.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198, 2016.

[Ding *et al.*, 2018] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. An improved sampler for bayesian personalized ranking by leveraging view data. In *WWW*, pages 13–14, 2018.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[Garcia-Molina *et al.*, 2011] Hector Garcia-Molina, Georgia Koutrika, and Aditya Parameswaran. Information seeking: convergence of search, recommendations, and advertising. *Communications of the ACM*, 54(11):121–130, 2011.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.

[He *et al.*, 2016a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[He *et al.*, 2016b] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.

[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, Xiaoyu Du, and Tat-Seng Chua. Adversarial personalized ranking for item recommendation. In *SIGIR*, 2018.

[Hornik, 1991] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[Liang *et al.*, 2016] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *WWW*, pages 951–961, 2016.

[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.

[Tay *et al.*, 2018] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*, pages 729–739, 2018.

[Wang *et al.*, 2015] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. Exploring implicit hierarchical structures for recommender systems. In *IJCAI*, pages 1813–1819, 2015.

[Wang *et al.*, 2017] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In *SIGIR*, pages 185–194, 2017.

[Wang *et al.*, 2018a] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*, pages 1543–1552, 2018.

[Wang *et al.*, 2018b] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *WSDM*, pages 619–627, 2018.

[Xue *et al.*, 2017] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, pages 3203–3209, 2017.

[Yu *et al.*, 2018] Wenhui Yu, Huidi Zhang, Xiangnan He, Xu Chen, Li Xiong, and Zheng Qin. Aesthetic-based clothing recommendation. In *WWW*, pages 649–658, 2018.

[Zhang *et al.*, 2014] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92, 2014.

[Zhang *et al.*, 2016] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete collaborative filtering. In *SIGIR*, pages 325–334, 2016.

[Zhang *et al.*, 2017] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. Joint representation learning for top-n recommendation with heterogeneous information sources. In *CIKM*, pages 1449–1458, 2017.