Outer Product-based Neural Collaborative Filtering

Xiangnan He¹, Xiaoyu Du¹², Xiang Wang¹, Feng Tian³, Jinhui Tang⁴ and Tat-Seng Chua¹
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 datasets 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 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., 2011]. As a typical paradigm of information push, recommender systems have become a core service and a major mon\-etization method for many customer-oriented systems [Wang et al., 2011]. As a typical paradigm of information push, recommender systems have become a core service and a major mon\-etization method for many customer-oriented systems [Wang et al., 2011]. As a typical paradigm of information push, recommender systems have become a core service and a major mon\-etization method for many customer-oriented systems [Wang et al., 2011]. As a typical paradigm of information push, recommender systems have become a core service and a major mon\-etization method for many customer-oriented systems [Wang et al., 2011]. As a typical paradigm of information push, recommender systems have become a core service and a major mon\-etization method for many customer-oriented systems [Wang et al., 2011]. As a typical paradigm of information push, recommender systems have become a core service and a major mon\-etization method for many customer-oriented systems [Wang et al., 2011].

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.,

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

We first present the Outer product based Neural Collaborative Filtering (ONCF) framework. We then elaborate on our proposed Convolutional 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 of matrix \( \mathbf{P} \), and vector \( \mathbf{p}_u \) denotes the \(u\)-th row vector in \( \mathbf{P} \). Let \( \mathcal{S} \) be a 3D tensor, then scalar \( s_{a,b,c} \) denotes the \((a, b, c)\)-th element of tensor \( \mathcal{S} \), and vector \( \mathbf{s}_{u,b} \) 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., \( g_{ui} \); 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 \( \mathbf{v}_u \) and \( \mathbf{v}_i \) be the feature vector for user \( u \) and item \( i \), respectively, we can obtain their embeddings \( \mathbf{p}_u \) and \( \mathbf{q}_i \) via:

\[
\mathbf{p}_u = \mathbf{P}^T \mathbf{v}_u, \quad \mathbf{q}_i = \mathbf{Q}^T \mathbf{v}_i, \tag{1}
\]

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
Interaction Map

learning model to generalize well on sparse data, whereas modeling the interaction of features, which only retains the original information in embeddings without modeling any correlation. Moreover, it has been recently shown that, modeling the interaction of features 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 \( g = f_\Theta (E) \), where \( f_\Theta \) denotes the model of hidden layers that has parameters \( \Theta \), and \( g \) is the output vector to be used for the final prediction. Technically speaking, \( f_\Theta \) 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 \( g \) and outputs the prediction score as: \( \hat{y}_{ui} = w^T g \), where vector \( w \) re-weights the interaction signal in \( g \). To summarize, the model parameters of our ONCF framework are \( \Delta = \{ P, Q, \Theta, w \} \).

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:

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

where \( \lambda_\Delta \) are parameter specific regularization hyperparameters to prevent overfitting, and \( D \) denotes the set of training instances: \( D := \{(u,i,j)|i \in Y_u^{-} \land j \notin Y_u^{-}\} \), where \( Y_u^{-} \) 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 \( w \) can control the magnitude of the value of \( \hat{y}_{ui} \) for all predictions. As a result, scaling up \( w \) 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 \( w \). 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 \( E \in \mathbb{R}^{K \times K} \), we can flat \( E \) 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 performance².

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 in hidden layer l is represented as a 2D matrix Eᵢ²: since we set the stride to 2, the size of Eᵢ² is half of its previous layer l − 1, e.g. E¹ = ∈ ℝ₃₂×₃₂ and E² ∈ ℝ₁₆×₁₆. All feature maps of Layer l can be represented as a 3D tensor Eᵢ².

Given the input interaction map E, we can first get the feature maps of Layer 1 as follows:

\[ E¹ = [e_{i,j,c}]_{32 \times 32 \times 32}, \quad \text{where} \]

\[ e_{i,j,c}^1 = \text{ReLU}(b_1 + \sum_{a=0}^{l} \sum_{b=0}^{l} e_{2i+a,2j+b} \cdot t_{1-a,1-b,c}^1), \quad (4) \]

where \( b_1 \) denotes the bias term for Layer 1, and \( T^1 = [t_{a,b,c}]_{2 \times 2 \times 32} \) is a 3D tensor denoting the convolution filter for generating feature maps of Layer 1. We use the rectifier 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 Eᵢ²⁺¹:

\[ E^{l+1} = [e_{i,j,c}^{l+1}]_{s \times s \times 32}, \quad \text{where} \quad 1 \leq l \leq 5, \quad s = \frac{64}{2^{l+1}}, \]

\[ e_{i,j,c}^{l+1} = \text{ReLU}(b_{l+1} + \sum_{a=0}^{l} \sum_{b=0}^{l} e_{2i+a,2j+b} \cdot t_{1-a,1-b,c}^{l+1}), \quad (5) \]

where \( b_{l+1} \) denotes the bias term for Layer \( l + 1 \), and \( T^{l+1} = [t_{a,b,c,d}]_{2 \times 2 \times 32 \times 32} \) denote the 4D convolution filter for Layer \( l + 1 \). The output of the last layer is a tensor of dimension 1 × 1 × 32, which can be seen as a vector and is projected to the final prediction score with a weight vector 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 E, each entry \( e_{x,y,c} \) encodes the second-order correlation between the dimension i and j. Next, each hidden layer l captures the correlations of a 2 × 2 local area³ of its previous layer \( l − 1 \). As an example, the entry \( e_{x,y,c}^1 \) 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 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 an 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?

³The size of the local area is determined by our setting of the filter size, which is subjected to change with different settings.
### Table 1: Top-k recommendation performance where $k \in \{5, 10, 20\}$.

<table>
<thead>
<tr>
<th></th>
<th>Gowalla</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HR@$k$</td>
<td>HR@$k$</td>
<td>NDCG@$k$</td>
<td>HR@$k$</td>
<td>NDCG@$k$</td>
<td>RI</td>
</tr>
<tr>
<td></td>
<td>$k = 5$</td>
<td>$k = 10$</td>
<td>$k = 20$</td>
<td>$k = 5$</td>
<td>$k = 10$</td>
<td>$k = 20$</td>
</tr>
<tr>
<td>ItemPop</td>
<td>0.2005</td>
<td>0.2705</td>
<td>0.3139</td>
<td>0.1099</td>
<td>0.1350</td>
<td>0.1591</td>
</tr>
<tr>
<td>MF-BPR</td>
<td>0.6284</td>
<td>0.7480</td>
<td>0.8422</td>
<td>0.4825</td>
<td>0.5214</td>
<td>0.5454</td>
</tr>
<tr>
<td>MLP</td>
<td>0.6359</td>
<td>0.7590</td>
<td>0.8535</td>
<td>0.4802</td>
<td>0.5202</td>
<td>0.5443</td>
</tr>
<tr>
<td>JRL</td>
<td>0.6605</td>
<td>0.7747</td>
<td>0.8561</td>
<td>0.5270</td>
<td>0.5615</td>
<td>0.5821</td>
</tr>
<tr>
<td>NeuMF</td>
<td>0.6794</td>
<td>0.7919</td>
<td>0.8602</td>
<td>0.5719</td>
<td>0.5860</td>
<td>0.5865</td>
</tr>
<tr>
<td>ConvNCF</td>
<td>0.7114</td>
<td>0.7936</td>
<td>0.8695</td>
<td>0.5244</td>
<td>0.5826</td>
<td>0.6019</td>
</tr>
</tbody>
</table>

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 and Gowalla.

**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-10 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/dxy-m/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.
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. 2016Y0240, and by the Natural Science Foundation of Heilongjiang Province of China (No.F2016002). Jinhui Tang is the corresponding author.

References


[He et al., 2018] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. Adversarial personalized ranking for item recommendation. In SIGIR, 2018.


