Proceedings of Machine Learning Research 101:726–741, 2019 ACML 2019

Convolutional Neural Collaborative Filtering with Stacked Embeddings

Liu Han M201773002@hust.edu.cn
Hailong Wu M201672856@hust.edu.cn
Nan Hu nanhu@hust.edu.cn
Binbin Qu binbinqu_hust@163.com
Huazhong University of Science and Technology, Wuhan, China

Abstract

Recommender System plays an important role in keeping people engaged with online services, and collaborative filtering is a main technique for recommendation. With the immense influence of deep learning, there is a growing interest in applying it to collaborative filtering. Existing methods have applied different ways to learn the user-item interaction function, however, most of these methods have limitation in modeling user-item correlations because they ignore the original user-item information and the large size of embeddings. In this work we propose Stacked Embedding Convolutional Neural Collaborative Filtering (SECNCF), a novel neural collaborative filtering architecture. The idea is to create a pedrail by stacking embeddings which are composed of user embedding, item embedding and latent factors. We apply convolutional neural network (CNN) above the pedrail layer to capture the local features of dimension correlations. This method is good at extracting rich local dimension correlations of embeddings and is scalable for modeling user-item interactions. Extensive experiments on three public accessible datasets show that our method makes significant improvement over the state-of-the-art methods.

Keywords: Collaborative Filtering, Convolutional Neural Network, Stacked Embedding, Recommender Systems

1. Introduction

In the era of information explosion, recommender systems have been applied to various online services to alleviate information overload, such as online advertisement, online news and social medias. In recommender systems, collaborative filtering (CF) infers a user's preference from not only her behavior data but also the behavior data of other users. As a kind of CF method, matrix factorization (MF) based methods (Rendle et al. (2009);He et al. (2016)) have become the mainstream of recommendation research and application due to the superior performance. The key designs in collaborative filtering are how to represent users and items and how to express the interactions between them. MF projects user and item to a shared low-dimension space and uses inner-product to express the interactions of user-items.

Despite the effectiveness of MF, we know that inner-product is a simple linear function which may not be sufficient to capture the complicated relationships of user-item interactions. So many research ideas have been devised to enhance MF's performance and efficiency. Those methods can be separated into two sorts: one is improving the model it-

self (Wang et al. (2015);Yu et al. (2018)), for example: DeepMF Xue et al. (2017) improved MF by applying deep neural networks to learn user and item representations; the other is improving the learning strategy (Rendle et al. (2009);Bayer et al. (2017);He et al. (2018b)), for example: BPR Rendle et al. (2009) learned MF in pairwise ranking perspective.

Among the neural network methods for collaborative filtering, neural matrix factorization (NeuMF) He et al. (2017) proposed to replace inner product with multiple-layer perceptron (MLP) to learn user-item interactions. Later it's prevalent to use MLP such as put a MLP above the element-wise product of user-item embedding. However, such designs suffer from limitations about expressing dimension correlations, especially for using element-wise product only. Although in theory, MLP can approximate any continuous function on the basis of the universal approximation theorem Hornik (1991), there is no practical guarantee to show we can effectively capture the correlations of dimensions based on current optimization methods. And then ConvNCF He et al. (2018a) proposed to put a deep convolutional network above the interaction map which is composed of the outer-product of user-item embeddings. It outperforms but still has the limitations that the embeddings have to be pretrained with MF_BPR Rendle et al. (2009) and the size of interaction map is determined by embedding size which is normally too big.

In this work, we propose a new neural network method Stacked Embedding Convolutional Neural Collaborative Filtering (SECNCF), and present a new structure called 'pedrail' which is a chain structure composed of stacked embeddings, these stacked embeddings link together to better capture embedding dimension interactions and it will be described in detail in section 4. This structure will combine different embeddings without losing their original information, meanwhile the size of interaction map is moderate and easy to extend which is different with the mentioned user-item embedding concatenation (He et al. (2017);He et al. (2018a)). Above the pedrail layer, we adopt a 2-dimensional convolutional neural network (CNN) to learn the local dimension correlations, and finally we can learn a mapping function from the output of CNN to get a better user-item interaction function. It is known that convolutional network is good at learning local features and the number of feature maps (kernels) can reveal multiple aspects of local dimension correlations. In this work, we focus on how to learn better user-item interaction function from the noisy implicit feedback data.

The main contributions of this work are as follows:

- We propose a new and extensible method named stacking embeddings to combine user embedding and item embedding.
- We propose a new neural collaborative architecture SECNCF, based on stacked embeddings, we adopt CNN to learn local dimension correlations, and the full dimension interactions through the final full mapping.
- We conduct extensive experiments on three public accessible datasets, the results demonstrate the effectiveness of SECNCF and achieve state-of-the-art performance.

2. Related Work

In recent years, rating prediction has been well solved by work on explicit feedback like ratings or reviews. And implicit feedback is largely investigated for item recommendation since

explicit feedback has the drawback of ignoring missing data. There are two main strategies to deal with missing data —— either treat all missing data as negative instances (Hu et al. (2008)) or sample negative instances from missing data (Rendle et al. (2009)). Ning and Karypis (2011) proposed SLIM and made a special case of Matrix Factorization(MF) which makes user use the original item and each item is represented by a linear combination of other items. Kabbur et al. (2013) extended SLIM (Ning and Karypis (2011)) to FISM to learn the latent factor matrix of two items and improves the HR metrics a lot. Wu et al. (2016) presented CDAE to make top-N recommendations using Denoising AutoEncoder and handling the preferences for items with noise.

Then MLP is prevalent in recommendation. He et al. (2017) proposed NCF which applies MLP on user-item embedding concatenation to learn user-item interaction function, adopted uniform negative sampling strategy and achieved state-of-the-art performance for item recommendation. Wang et al. (2017) extended the NCF model to cross-domain recommendation, further demonstrating the effectiveness of explicitly modeling user-item interaction function. Xue et al. (2017) showed that the onehot identifier can be replaced with explicit user-item interaction vector to retain the user-item interaction patterns and proved the usefulness of explicit interaction data. CNNs are widely used for feature representation learning for image Yu et al. (2018), text Kim et al. (2016), and audio Van den Oord et al. (2013) in recommender systems. CNN can also be directly applied to collaborative filtering, for example: He et al. (2018a) improved NCF and proposed ConvNCF which uses outer-product to replace inner-product and applied CNN to learn high-order correlations among embedding dimensions. This is similar with our SECNCF but our model is more flexible and easy to extend.

3. Preliminaries

In this section, we will first formalize the problem and present our data processing method. Later we will give a brief review of MF.

3.1. Problem Definition

As seen in existing researches on collaborative filtering (CF), comparing with explicit feedback (likes ratings or reviews), most works focus on implicit feedback which can reflect users' preference from some potential information (likes record of clicking, visiting, comments) automatically. However, implicit feedback learning is more challenging because of the unobservability of users' preference and some noisy implicit data.

To explore this problem, this paper takes implicit feedback data as train and test datasets to complete the recommendation task. In a standard recommendation task, we have M users, N items, and user-item interaction matrix $R \in R^{M \times N}$. Here $R_{u,i}$ (element in matrix R) could represent the interest of user u for item i. We define the implicit feedback matrix $Y \in R^{M \times N}$ for user and item as:

$$y_{u,i} = \begin{cases} 1, \text{ if interaction of (u,i) is observed.} R_{u,i} \neq 0; \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

Here 1 indicates that there is interaction between user u and item i, however, 0 does not mean that user u dislikes item i, it just shows user u has not seen item i yet. This brings challenges for implicit feedback task because the unobserved data may be just missing data and provides noisy signals.

The target of using implicit feedback data for recommendation is to predict the unobserved score in Y which can be abstracted as learning $y_{u,i} = f(u, i|\Theta)$, in which Θ represents parameters of the model and f defines a function that maps model parameters to a predicted score.

In order to learn the model parameters, we should create an objective function and optimize it with training data. There exists two kinds of commonly used objective function for recommendation: pointwise function and pairwise function. For pointwise method, they usually transform the target problem to a regression problem to minimize the squared loss between the target score $y_{u,i}$ and the predicted score $\hat{y}_{u,i}$. As for the noisy unobserved signals, they commonly set all of the unobserved data as negative instances or sample negative data from it. Pairwise methods optimize the objective function by maximizing the margin of observed entries and unobserved ones (negative instances). Similar to existing model-based methods, our model builds a mapping function f to estimate $y_{u,i}$, and takes pointwise as the objective function. Meanwhile, it's easy to extend to pairwise setting with BPR Rendle et al. (2009), and this would be tried in the future research.

3.2. Matrix Factorization

Proposed by Koren et al. (2009) at Netflix contest, Matrix Factorization (MF) becomes prevalent in collaborative filtering due to its high accuracy and scalability. In general, MF can learn low-dimension latent vector representation from user-item interaction matrix, and can predict the unobserved instances (ratings, visited or not...etc.). If we represent user u, item i with p_u and q_i, MF can estimate their interactions through inner product:

$$y_{u,i} = f(u, i|p_u, q_i) = p_u^T q_i = \sum_{k=1}^{K} p_{u,k} q_{i,k}. \tag{2}$$

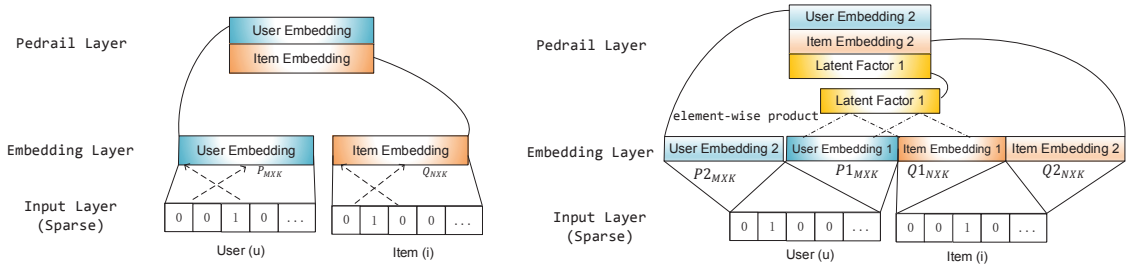
Here K is the dimension of latent space. MF assumes that dimensions of latent vector are independent, and we can associate them with the weight. In this point of view, MF is a kind of linear latent factor model. Here we present you the commonly used objective function in MF for score prediction:

$$\mathcal{L} = arg \min_{P,Q} \sum_{u=1}^{M} \sum_{i=1}^{N} I_{u,i} \sigma(y_{u,i}, \hat{y}_{u,i}) + \lambda_p \|P\|_F^2 + \lambda_q \|Q\|_F^2. \tag{3}$$

Here σ denotes for loss function of $y_{u,i}$ and $\hat{y}_{u,i}$, $I_{u,i}$ indicate whether there is observed instance for (u,i) or not and P,Q are user and item latent factor matrixes. In addition, $\|P\|_F^2$ and $\|Q\|_F^2$ denote the Frobenius norm of the matrix. λ_p and λ_q are regularization parameters to alleviate overfitting.

4. Proposed Methods

We will first talk about the idea of stacked embedding. Then we will give you the details of our proposed CNN architecture SECNCF and analyze why pedrail layers can obtain better embedding dimension correlations. Last we will present you with two model instances of SECNCF: SEC2NCF and SEC3NCF.



(a) The pedrail-2 structure.

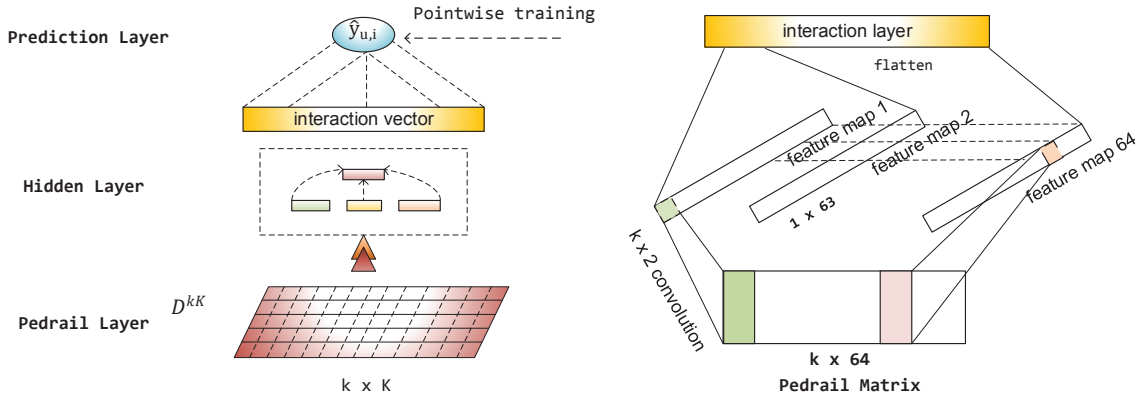
(b) The pedrail-3 structure.

Figure 1: These two pictures demonstrate two instances for the pedrail structure. (a) $P_{M\times K}$ and $Q_{M\times K}$ denotes the embedding matrix of user and item respectively ; (b) $P^1_{M\times K}$ and $Q^1_{M\times K}$ denote one pair of user and item embedding matrix and $P^2_{M\times K}$ and $Q^2_{M\times K}$ denote another pair.

4.1. Stacked Embeddings

As is mentioned before, Stacking embedding can not only combine different latent factors but also make the size of stacked embeddings more flexible. Stacking is an operation that combines different embeddings together in the same direction which makes the pedrail consist of different latent factors, and then we can learn the dimension interactions better. The stacked embeddings stack user embedding, item embedding, and the latent factors (they may be transformed from explicit ratings or other information) by mathematical computation in formula (4). we can stack some $1 \times K$ embeddings horizontally following their bigger dimension K and get a $k \times K$ embedding. After stacking, we obtain a 'pedrail' with the size of $k \times K$, here k denotes the number of stacked embeddings and K denotes the dimension of embeddings.

we present two instances of the pedrail structure with k equals 2 and 3 in Figure 1. In Figure 1 (a) it shows $2\times K$ pedrail stacks user embedding and item embedding. Figure 1 (b) presents $3\times K$ pedrail stacks user embedding, item embedding and the element-wise product of another user-item embeddings. As for the element-wise product result, we choose it as the latent factor stacked in the pedrail because it is efficient and easy to comprehend, besides, it's widely used in neural collaborative filtering. It is a representative latent factor which will help to prove the validity of our model. Meanwhile, we can stack more information to form a $4 \times K$ pedrail or even higher k value if we have latent factors like information of comments or description of items. It is challenging but expensive because higher k value means that the pedrail is more complicated and we need more space to store it and more time to train the model with more datasets. So we should adjust the size of the pedrail within reasonable limits.



(a) Stacked-Embedding-Convolutional Neural Collaborative Filtering model

(b) The applied convolutional structure in SECNCF architecture.

Figure 2: Model structures. (a) gives the general structure of our proposed SECNCF architecture; (b) presents the specific convolutional structure which we apply in this work.

4.2. Stacked Embedding Convolutional Neural Collaborative Filtering

As shown in Figure 2 (a), SECNCF model consists of the pedrail layer which contains latent factors, the hidden layer used to deal with embedding dimension correlations and the final prediction layer which is targeted for the score estimation.

Pedrail Layer. In basic collaborative filtering setting, user ID and item ID are used as the only input information and our model will follow this setting. From the IDs we can get user embedding $P_u \in R^K$ and item embedding $Q_i \in R^K$ where K denotes the dimension of embeddings. Representing $k \times K$ pedrail with $D_{ui}^{k\times K} \in R^{k\times K}$, we only formulate $2 \times K$ pedrail and $3 \times K$ pedrail used in the experiments as follows:

$$\begin{aligned} D_{ui}^{2\times K} &= s(P_u, Q_i), \\ mult &= P_u^{'} \odot Q_i^{'}, \\ D_{ui}^{3\times K} &= s(P_u, Q_i, mult). \end{aligned} \tag{4}$$

Here P_u and Q_i denote embeddings of user u and item i respectively. $P_u^{'}$ and $Q_i^{'}$ denote another pair of user u and item i embeddings. $\odot$ denotes element-wise product. s denotes the stacking operation, and its formulation is as follow:

$$s(A_1, A_2, ..., A_n) = \begin{bmatrix} A_1 \\ A_2 \\ ... \\ A_n \end{bmatrix} \tag{5}$$

To better understand the stacking operation, we give an example :the user embedding P_u is $\begin{bmatrix} 0.21 & 0.14 & 0.10 & 0.09 \end{bmatrix}$ and the item embedding Q_i is $\begin{bmatrix} 0.18 & 0.24 & 0.30 & 0.10 \end{bmatrix}$, and then $\mathrm{s}(P_u,Q_i) = \begin{bmatrix} 0.21 & 0.14 & 0.10 & 0.09 \\ 0.18 & 0.24 & 0.30 & 0.10 \end{bmatrix}$. This is the fundamental structure of our proposed

SECNCF recommendation model. The advantages of applying stacking to latent factors are summarized as follows:

- Comparing with Matrix Factorization (MF), this structure encodes more signals and can be easily extended. For example, we stack element-wise product in SEC3NCF which is also a latent factor in MF, and we can incorporate more latent factors like comments or side information of users and items;
- Comparing with simple embedding concatenation, this structure can extract more expressive information of dimension correlations with CNN. For example, our model is more scalable for encoding more information and simpler to optimize without more constraints like the outer-product based method ConvNCF He et al. (2018a).

Hidden Layer. Above the pedrail layer is the hidden layer for extracting valuable correlations of latent factor dimensions. We can capture more useful signals with special hidden layer designs where we reach $v_{hidden} = f_\Theta(D_{ui}^{kK})$, here f_Θ denotes the hidden layer with Θ as parameters and v_{hidden} denotes the output vector of hidden layer which will be transferred to prediction layer. It's worth noting that the hidden layer needs to be specially designed to capture valuable signals of dimension correlations, in our model we only consider CNN and we will try other models in future research. In section 4.3, we will show you how to apply CNN to extract valuable interaction features on dimensions.

Prediction Layer. Receiving latent vector v_{hidden} as input, which successfully captures correlations over dimensions, prediction layer applies $\hat{y}_{u,i} = a_{out}(w^T v_{hidden})$ to make predictions, w are parameters of prediction layer which denotes weights of elements of v_{hidden} and a_{out} denotes the activation function of the prediction layer which is generally a sigmoid function ($\sigma(x) = 1/(1 + e^{-x})$) for implicit feedback tasks. In summary, the whole parameters of SECNCF are $\{\Theta, P, Q, w\}$.

Learning SECNCF for Recommendation. The traditional pointwise methods to learn the parameters is squared loss just like:

$$\mathcal{L}_{squared} = \sum_{(u,i)\in\mathcal{Y}} w_{u,i}(y_{u,i} - \hat{y}_{u,i})^2. \tag{6}$$

but we find that it is not totally appropriate for the implicit data because $y_{u,i}$ is 1 or 0 for the implicit data which means whether user-item interaction exists, and squared loss assumes that $y_{u,i}$ is generated from Gaussian distribution.

We take a probability function (logistic or probit function) for learning pointwise and optimize our SECNCF models settings where we use 1 represents for the existence of user-item interactions and 0 otherwise. To achieve this goal, we should restrict the final prediction score $\hat{y}_{u,i}$ in the range of [0, 1]. According to the parameter settings above, we will offer you the likelihood function:

$$\mathcal{P}(\mathcal{Y}|\Theta, P, Q, w) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{u,i}^{y_{u,i}}(1 - \hat{y}_{u,i})^{(1-y_{u,i})}. \tag{7}$$

Applying the negative logarithm to the likelihood function we can get:

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{Y}} y_{u,i}\log(\hat{y}_{u,i}) + (1 - y_{u,i})\log(1 - \hat{y}_{u,i}). \tag{8}$$

This is the objective function for SECNCF to optimize with stochastic gradient descent (SGD) method. As we treat the implicit feedback task as a binary classification problem, we take the binary cross entropy loss as the objective function. Here for the negative instances, we uniformly sample them from the unobserved instances ($y_{u,i} = 0$) as the data is too big. Clearly better sample strategy could lead to better results, such as considering the popularity of items.

Motivation of taking CNNs. According to the above settings, MLP can also be applied for hidden layers (by flatting the pedrail layer). However, it would loss the signals of local dimension correlations if we flat the $k \times K$ pedrail, which is the same as NCF He et al. (2017) setting. Although MLP can approximate any continuous function in theory, the performance of MLP is actually data driven which means it may easily lead to suboptimal results on real world sparse dataset. Moreover, MLP has a large number of parameters especially for deeper networks, which practically increases difficulties for normal machines. These problems can be alleviated by utilizing CNN. On one hand, CNN is proficient in handling local relationships and can learn multiple aspects of local unit interactions with different kernels. On the other hand, CNN can be greatly accelerated by taking advantage of parallel computation of GPUs which makes the model practically feasible.

4.3. SEC2NCF and SEC3NCF

Since the pedrail matrix size is $k \times K$ in SECNCF model, we uniformly apply a 2-dimensional convolution above the pedrail layer to learn local dimension correlations with kernel size $k \times 2$ (k denotes the number of latent factors, 2 in SEC2NCF and 3 in SEC3NCF). The column size of the kernel can be any size between [2, K] than 2 only. We can also follow Kim (2014) which applies several kernel sizes to extract features of different area. In this work, we have tried to combine kernel size $k \times 2$ with $k \times 3$ to extract the features together and the performance indicates this is a good idea for future exploration.

Figure 2 (b) shows how we apply convolutions on the pedrail matrix. As is shown in the figure, the pedrail layer is represented by a $k \times 64$ matrix where 64 is the embedding size. Here we only talk about how we apply convolutions on the pedrail layers for SECNCF. Convolutional networks are too complicated (kernel sizes, strides, and paddings, etc.) to give a systematic formulation, thus we only formalize the convolutions we used in SEC2NCF and SEC3NCF. In fact, we can apply various CNN structures so long as we carefully handle the structures and parameters. However, as we talked in Section 4.2, deeper CNN could lead to poor performance which is similar with MLP He et al. (2017) .

We apply 2×2 and 3×2 kernel size for SEC2NCF and SEC3NCF respectively with 64 feature maps. Here we only give you the formula of convolutions for SEC2NCF since SEC3NCF is similar. The pedrail matrix of SEC2NCF is represented as $D \in R^{2 \times 64}$; since we set the stride to 1, the size of the feature maps is 1×63 which demonstrates one aspect of local dimension correlations. Taking $2 \times 2 \times 64$ convolution where 64 is the number of convolution kernels, we have the resulting tensor representation of the feature maps

$T \in R^{1\times 63\times 64}$ which is formulated as follows:

$$T = [t_{i,j,c}]_{1\times 63\times 64}, where$$
$$t_{i,j,c} = ReLU(\sum_{d=0}^{1}\sum_{e=0}^{1} D_{i+d,j+e} \cdot \underbrace{c_{1-d,1-e,c}}_{\text{convolution kernel}} + b). \tag{9}$$

Here $t_{i,j,c}$ denotes the element of feature maps, b denotes the bias term in convolutional layer and $\mathcal{C} = [c_{d,e,c}]_{2\times 2\times 64}$ is a 3D tensor represents the convolution kernel. The output tensor T will be flatted to a vector and sent to prediction layer. For the CNN model, we require $2 \times 2 \times 64 + (64 - 1) \times 64 = 4288$ parameters in which $2 \times 2 \times 64$ represents for the parameters of convolutional networks and $(64 - 1) \times 64$ represents for the number of parameters in prediction layer. However, even for the simple two-layer MLP with dimensions 64×32 we require $2 \times 64 \times 64 \times 32 = 262144$ parameters (we ignored the bias term for simplicity), let alone more layers. We can see that the number of parameters in CNN is many orders of magnitude smaller than that in MLP, which makes SEC2NCF (SEC3NCF has the same magnitude of parameters) more stable and more generalizable.

5. Experiment

In this section, to demonstrate the effectiveness of our proposed SECNCF methods, we carry out experiments to answer the following 4 research questions (RQ).
RQ1. Do our proposed SECNCF methods outperform the state-of-the-art implicit feedback collaborative filtering methods?
RQ2. Are the proposed stacking embedding methods and the CNN layer helpful in learning user-item interaction function?
RQ3. How does the negative sampling ratio affect the performance of SECNCF model?
RQ4. How do the hyperparameters (i.e. the number of feature maps) influence the performance of SECNCF?

5.1. Experiment Settings

Our experiment environment is ordinary server with 1 CPU (i7-5930k), 2 GPU (GTX 1080) and 64G RAM. We use Keras (using Tensorflow as backend) as the deep learning framework.

Dataset Description. We conduct experiments on three public available Amazon product ratings datasets[1]: Amazon ratings Movies and TV (AMT),Amazon ratings CDs and Vinyl (ACV), and Amazon ratings Kindle Store (AKS).

All these datasets are released by Amazon for research purpose. Each sample instance in the datasets consists of user ID, item ID, rating (range in [1, 5]) and timestamp. Following the data preprocessing method in implicit feedback recommendation methods like (He et al. (2017); He et al. (2018a)), we remove the user instances where user's rating records are less than 20. The statistics of the preprocessed datasets are summarized in Table 1.

1. http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/

Table 1: Statistics of Datasets

Dataset	Interaction#	User#	Item#	Density
ACV	964681	15616	290296	0.0213%
AKS	745293	14813	185028	0.0272%
AMT	953682	16141	116565	0.0507%

Evaluation Protocols. For each dataset, we conduct leave-one-out strategy which has been widely used before (Bayer et al. (2017);He et al. (2016);He et al. (2018a);He et al. (2017)). We held out the latest instance for each user as test set and left the remaining instances for training. Since the number of items in each dataset is very large, we randomly sample 99 unobserved instances for each user and then pair with the test set. We make predictions for these 100 user-item interactions with each methods. To evaluate the results, we adopt Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) metrics which is same as He et al. (2017). HR@K represents whether the testing item appears in the top-k ranking list (1 for yes, 0 for no) and NDCG@K gives higher score to the item if it is higher ranking in the top-k list. We evaluate these two metrics for each user and report the average score after convergence.

Baselines. We compare our proposed methods (SEC2NCF and SEC3NCF) with the following methods:

- **ItemPop.** Items are ranked by their popularity which is the number of interactions. It is always taken as a benchmark algorithm for recommendation.

- **JRL.** Zhang et al. (2017) places a MLP above the element-wise product of user-item embeddings to learn user-item interaction function which is different with GMF as JRL uses multiple hidden layers.

- **ConvNCF-p.** He et al. (2018a) generates interaction map using outer-product of user and item embeddings, places a deep convolutional network to learn user-item interaction function. The paper proposes to optimize the model in pairwise setting but here we train it in pointwise setting and we name it ConvNCF-p considering the performance.

- **GMF.** He et al. (2017) replaces the sum operation above the element-wise product of embeddings in MF with a linear mapping.

- **MLP.** He et al. (2017) concatenates user embedding and item embedding to feed to the multiple-layer perceptron (MLP) for learning user-item interaction function.

- **NeuMF.** He et al. (2017) is a state-of-the-art method which combines hidden layer of GMF and MLP to learn the interaction function.

Since our proposed method focuses on learning the user-item interactions, there are some other recommendation methods such as SLIM Ning and Karypis (2011), FISM Kabbur et al. (2013), CDAE Wu et al. (2016) which are item-item models. We leave out the comparison

with them as the performance may be influenced by users' personalization model. And It's important to note that we do not report the results of BPR Rendle et al. (2009) because these datasets are too sparse and small and the performance results are bad.

Table 2: Top-k recommendation results of *Hit Rate* where $k \in \{5, 10, 20\}$.

	HR @ k								
	ACV			**AKS**			**AMT**		
	k=5	k=10	k=20	k=5	k=10	k=20	k=5	k=10	k=20
ItemPop	0.3312	0.4382	0.5392	0.2654	0.3793	0.5182	0.4620	0.6029	0.7303
ConvNCF-p	0.4107	0.5165	0.6210	0.4774	0.5869	0.6949	0.5241	0.6531	0.7576
JRL	0.4570	0.5393	0.6181	0.5042	0.6176	0.7004	0.5816	0.7009	0.7915
MLP	0.4631	0.5489	0.6407	0.5437	0.6333	0.7156	0.5642	0.6821	0.7840
GMF	0.5131	0.5931	0.6604	0.5763	0.6512	0.7194	0.6001	0.7163	0.8047
NeuMF	0.5008	0.5815	0.6561	0.5689	0.6508	0.7213	0.5857	0.7043	0.7935
SEC2NCF	0.5129	0.6103	0.6789	0.5783	0.6657	0.7410	0.6007	0.7182	0.8159
SEC3NCF	**0.5455***	**0.6454***	**0.7490***	**0.6028***	**0.6968***	**0.7809***	**0.6112***	**0.7352***	**0.8233***

Parameter Settings. We randomly sample a training instance for each user as the validation set to tune the hyperparameters and then we make predictions on the test set. We evaluate SEC2NCF and SEC3NCF in different parameter settings. For all the neural collaborative filtering (NCF) methods, we optimize with Equation 8, we sample 4 negative instances for each positive instance in default. To make a fair comparison, the embedding size is setted to 64 for all methods and initializing the parameters with Gaussian Distribution (0 for mean and 0.01 for standard deviation). We optimize all the models with mini-batch (batch size is 256) Adam Kingma and Ba (2014) (learning rate is 0.001) pointwisely.

For ConvNCF-p, we adopt the same setting with He et al. (2018a) and the only difference is that we train it with pointwise loss, because the datasets are small and sparse which lead to bad performance in pairwise setting . We use four layer MLP and apply different nonlinear activation functions (relu, elu and selu, etc.) for JRL. We adopt the same structures in He et al. (2017) for MLP and NeuMF, and test different embedding sizes. We find that for these three datasets, setting embedding size to 64 achieves the best results and it consists with the finding in He et al. (2017) that larger MLP tower could lead to overfitting. We also evaluate NeuMF with MLP pretraining and find it makes little improvement. we train all the models on these three datasets and report the best results for all these models.

5.2. Performance Comparison (RQ1)

Table 2 shows the top-k recommendation results of HR@k for SECNCF together with the compared methods and Table 3 shows the corresponding results of NDCG@k. Here k is setted to 5, 10 and 20 and the negative sampling ratio is 4 for all the experiments. The experimental results show that:

- SEC3NCF achieves the best performance in general and dramatically outperforms state-of-the-art methods. SEC2NCF also achieves good results which proves the effectiveness of our proposed methods.

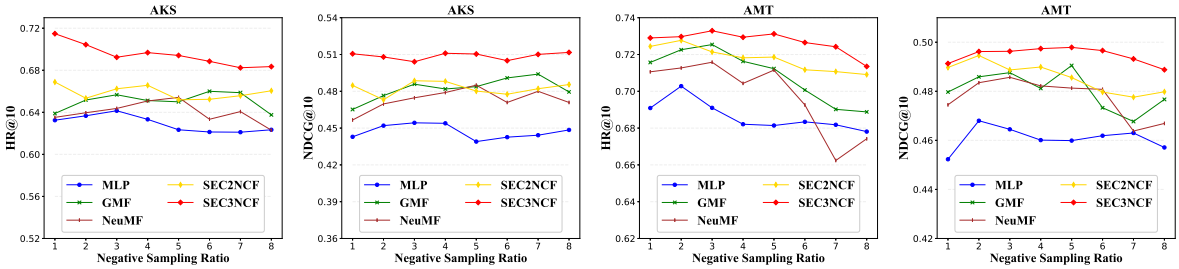
Table 3: Top-k recommendation results of *Normalized Discounted Cumulative Gain* where $k \in \{5, 10, 20\}$.

	NDCG @ k								
	ACV			**AKS**			**AMT**		
	k=5	k=10	k=20	k=5	k=10	k=20	k=5	k=10	k=20
ItemPop	0.2381	0.2727	0.2983	0.1843	0.2210	0.2560	0.3257	0.3713	0.4036
ConvNCF-p	0.3028	0.3370	0.3634	0.3580	0.3935	0.4209	0.3842	0.4262	0.4528
JRL	0.3590	0.3857	0.4056	0.3861	0.4225	0.4536	0.4348	0.4737	0.4968
MLP	0.3612	0.3891	0.4122	0.4249	0.4539	0.4747	0.4219	0.4601	0.4861
GMF	0.4097	0.4356	0.4527	0.4576	0.4819	0.4993	0.4495	0.4812	0.5082
NeuMF	0.3998	0.4260	0.4449	0.4524	0.4790	0.4969	0.4436	0.4822	0.5049
SEC2NCF	0.4071	0.4403	0.4554	0.4597	0.4881	0.5072	0.4516	0.4899	0.5147
SEC3NCF	**0.4320***	**0.4643***	**0.4904***	**0.4806***	**0.5110***	**0.5323***	**0.4590***	**0.4979***	**0.5213***

- ItemPop is the only method not to make personalized recommendation. It performs the worst in general which indicates that we should learn user preference explicitly rather than just recommend items with high popularity.
- ConvNCF-p performs much better than ItemPop which proves the importance of learning user preference explicitly. However, the results are worse than MLP. We think the reasons are twofold: the deep convolutional network in ConvNCF-p is hard to optimize with the sparse dataset, and the interaction map from the outer-product of user-item embeddings adds too many constraints to the embeddings like pretraining process.
- JRL performs worse than MLP on ACV and AKS but better on AMT. Judging by the fact that AMT is denser than ACV, AKS and the findings in He et al. (2018a) (JRL outperforms MLP in general), we think that explicitly modeling the correlations of embedding dimensions can learn user-item interaction function for denser dataset better.
- NeuMF outperforms MLP but is almost completely defeated by GMF which indicates that GMF is a simple yet powerful prediction model. NeuMF does not achieve expected results because the MLP part is hard to optimize in such sparse settings. As a simple but powerful method, GMF performs well on these three datasets and shows great potential on other tested datasets which indicates that GMF can be a strong baseline algorithm for recommendation.

5.3. Efficacy of Stacked Embedding and CNNs (RQ2)

From Table 2 and Table 3 we can see that comparing with other methods, the performance of SEC2NCF is only a little worse than GMF with top-5 metrics for ACV dataset, and is better than all the baselines in all metrics. Comparing the experimental results of SEC2NCF with MLP (without element-wise product), SEC3NCF with ConvNCF-p and NeuMF (with element-wise product), we can find that the model consists of Staked Embedding and CNNs can learn better user-item interaction function.



(*a*) AKS—HR@10 (*b*) AKS—NDCG@10 (*c*) AMT—HR@10 (*d*) AMT—NDCG@10

Figure 3: Experiment results of NCF methods on HR@10 and NDCG@10 *w.r.t.* the Negative Sampling Ratio.

Comparing SEC2NCF with MLP we can find that utilizing CNN to extract multiple aspects of local dimension relations can greatly improve model performance. Comparing SEC3NCF with ConvNCF-p (both have CNNs and element-wise product) we can learn the importance of stacked embedding structure and latent factors. Finally, comparing SEC3NCF with NeuMF we can learn the importance of organizing those embeddings (stacking or concatenating), and how to learn the interactions (using CNN instead of MLP). The comparisons of the model structures and experimental results show that combining stacked embedding with CNN is powerful which also demonstrates the efficacy of our model.

5.4. Influence of negative sampling ratio (RQ3)

Figure 3 shows the experimental results of our methods with different negative sampling ratio. We do not present the results for dataset ACV due to the space limitation and they are similar with the showing results.

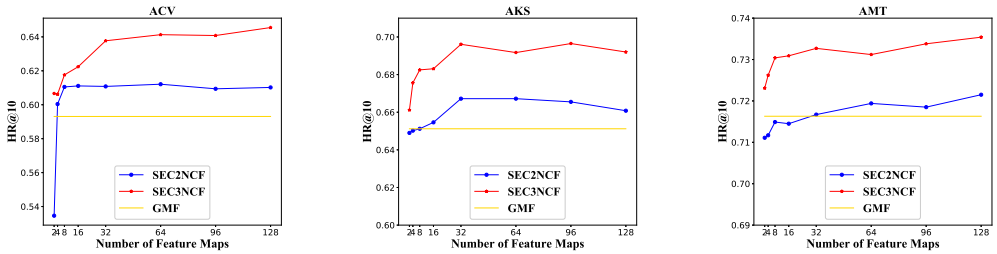
From the figure, we can see that for SEC3NCF the results of NDCG@10 on both datasets are very stable and the results of HR@10 are steep on both sides, but they tend to be smooth in general which means this method performs stable for top-K recommendation. SEC2NCF shows similar trend to SEC3NCF but the results of NDCG@10 on AMT become worse when the negative sampling ratio is bigger than 5, it means the noisy signals of negative samples could deteriorate the performance. On the other hand, SEC2NCF and SEC3NCF achieve the best results of HR@10 on AKS and AMT when negative sampling ratio is one (different on ACV) which indicates that we should tune the ratio of negative sampling for different datasets to obtain the best results.

NeuMF, MLP and GMF achieve the best results when the ratio is in [2, 5] which is in agreement with He et al. (2017). Besides, we can see that the results of NeuMF on AMT tend to be unstable when the ratio is bigger than 5 from the figure. To sum up, we find that SEC3NCF not only achieves best results with various negative sampling ratios but also performs very stable.

5.5. Hyperparameter Research (RQ4)

We compare the HR@10 metric results of SEC2NCF with SEC3NCF using different number of feature maps in Figure 4. Due to the space limitation we removed results of NDCG@10

metric which is consistent with HR@10. We set negative sampling ratio to 4 for all the experiments in the figure and present the results of GMF for comparison. Note that in order to make the figure clearer we remove the results that number of feature maps F equals to 1, but the results of $F = 1$ and $F = 2$ are very close so this does not influence the analysis of experimental results.



(a) ACV—HR@10 (b) AKS—HR@10 (c) AMT—HR@10

Figure 4: Experiment results of HR@10 for SEC2NCF and SEC3NCF *w.r.t.* the number of feature maps. We also show the corresponding results of GMF for comparison.

From the figure we can see that:

1. SEC3NCF outperforms GMF in all the experiments which means the element-wise product in pedrail layer is important; SEC2NCF outperforms GMF with enough number of feature maps which shows its features are not enough as SEC3NCF and we need more feature maps to capture different aspects of dimension correlations;
2. The results of SEC2NCF and SEC3NCF are consistent: they tend to be stable when $F > 8$ and they are stable when $F > 32$. This shows the great generalization capacity of CNNs since dramatically increasing the number of feature maps does not lead to overfitting. It means our proposed SECNCF model is suitable for practical usage;
3. The results of SEC3NCF is more stable than SEC2NCF when $F < 8$. The stable performance of SEC3NCF with small number of feature maps shows we can better learn user-item interaction function with stacking more latent factors.

6. Conclusion and Future Work

We propose a new neural collaborative architecture named SECNCF in this work. To learn a better user-item interaction function, we specially design a pedrail structure with stacked embeddings and apply a convolutional network, in which the pedrail can be extended easily and the convolutional network can learn local dimension correlations better. To demonstrate the capability of this architecture, we offer two model instances: SEC2NCF which only employs user embedding and item embedding, and SEC3NCF which stacks one more embedding, the element-wise product of another user-item embeddings. Extensive experiments on three public datasets demonstrate the superior capability of our model in top-K recommendation problems.

In future work, we will consider incorporate explicit ratings and other information of user/item into the architecture with a linear or nonlinear mapping. We will also investigate

the effect of combining multiple convolutional networks with different kernel-size like Kim (2014) and we will try to improve the performance with model pretraining. In addition, we will extend SECNCF into content-based recommendation scenarios where we have much more information than just IDs.

References

Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1341–1350. International World Wide Web Conferences Steering Committee, 2017.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558. ACM, 2016.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912*, 2018a.

Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 355–364. ACM, 2018b.

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

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

Santosh Kabbur, Xia Ning, and George Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 659–667. ACM, 2013.

Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 233–240. ACM, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*, pages 497–506. IEEE, 2011.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

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

Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 185–194. ACM, 2017.

Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162. ACM, 2016.

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.

Wenhui Yu, Huidi Zhang, Xiangnan He, Xu Chen, Li Xiong, and Zheng Qin. Aesthetic-based clothing recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 649–658. International World Wide Web Conferences Steering Committee, 2018.

Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1449–1458. ACM, 2017.