CoNet: Collaborative Cross Networks for Cross-Domain Recommendation

Guangneng Hu, Yu Zhang, Qiang Yang
Department of Computer Science and Engineering, Hong Kong University of Science and Technology
njuhgn@gmail.com,yu.zhang.ust@gmail.com,qyang@cse.ust.hk

ABSTRACT

The cross-domain recommendation technique is an effective way of alleviating the data sparse issue in recommender systems by leveraging the knowledge from relevant domains. Transfer learning is a class of algorithms underlying these techniques. In this paper, we propose a novel transfer learning approach for cross-domain recommendation by using neural networks as the base model. In contrast to the matrix factorization based cross-domain techniques, our method is deep transfer learning, which can learn complex user-item interaction relationships. We assume that hidden layers in two base networks are connected by cross mappings, leading to the collaborative cross networks (CoNet). CoNet enables dual knowledge transfer across domains by introducing cross connections from one base network to another and vice versa. CoNet is achieved in multi-layer feedforward networks by adding dual connections and joint loss functions, which can be trained efficiently by back-propagation. The proposed model is thoroughly evaluated on two large real-world datasets. It outperforms baselines by relative improvements of 7.84% in NDCG. We demonstrate the necessity of adaptively selecting representations to transfer. Our model can reduce tens of thousands training examples comparing with non-transfer methods and still has the competitive performance with them.

KEYWORDS

Recommender Systems; Collaborative Filtering; Transfer Learning; Deep Learning

ACM Reference Format:
Guangneng Hu, Yu Zhang, Qiang Yang. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *2018 ACM Conference on Information and Knowledge Management (CIKM'18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271684

1 INTRODUCTION

Collaborative filtering (CF) approaches, which model the preference of users on items based on their past interactions such as product ratings, are the corner stone for recommender systems. Matrix factorization (MF) is a class of CF methods which learn user latent factors and item latent factors by factorizing their interaction matrix [20, 28]. Neural collaborative filtering is another class of CF methods which use neural networks to learn the complex user-item interaction function [4, 9, 12]. Neural networks have the ability to learn highly nonlinear function, which is suitable to learn the complex user-item interaction. Both traditional matrix factorization and neural collaborative filtering, however, suffer from the cold-start and data sparse issues.

One effective solution is to transfer the knowledge from relevant domains and the cross-domain recommendation techniques address such problems [1, 2, 22, 33]. In real life, a user typically participates several systems to acquire different information services. For example, a user installs applications in an app store and reads news from a website at the same time. It brings us an opportunity to improve the recommendation performance in the target service (or all services) by learning across domains. Following the above example, we can represent the app installation feedback using a binary matrix where the entries indicate whether a user has installed an app. Similarly, we use another binary matrix to indicate whether a user has read a news article. Typically these two matrices are highly sparse, and it is beneficial to learn them simultaneously. This idea is sharpened into the collective matrix factorization (CMF) [38] approach which jointly factorizes these two matrices by sharing the user latent factors. It combines CF on a target domain and another CF on an auxiliary domain, enabling knowledge transfer [31, 48]. CMF, however, is a shallow model and has the difficulty in learning the complex user-item interaction function [9, 12]. Moreover, its knowledge sharing is only limited in the lower level of user latent factors.

Motivated by benefitting from both knowledge transfer learning and learning interaction function, we propose a novel deep transfer learning approach for cross-domain recommendation using neural networks as the base model. Though neural CF approaches are proposed for single domain recommendation [12], there are few related works to study knowledge transfer learning for cross-domain recommendation using neural networks. Instead, neural networks have been used as the base model in natural language processing [5, 46] and computer vision [8, 27, 47]. We explore how to use a neural network as the base model for each domain and enable the knowledge transfer on the entire network across domains. Then a few questions and challenges are raised: 1) What to transfer/share between these individual networks for each

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271684

domain? 2) How to transfer/share during the learning of these individual networks for each domain? and 3) How is the performance compared with single domain neural learning and shallow cross-domain models?

This paper aims at proposing a novel deep transfer learning approach by answering these questions under cross-domain recommendation scenario. The usual transfer learning approach is to train a base network and then copy its first several layers to the corresponding first layers of a target network with fine-tuning or parameter frozen [47]. This way of transferring has possibly two weak points. Firstly, the shared-layer assumption is strong in practice as we find that it does not work well on real-world cross-domain datasets. Secondly, the knowledge transfer happens in one direction, i.e., only from source to target. Instead, we assume that hidden layers in two base networks are connected by dual mappings, which do not require them to be identical. We enable dual knowledge transfer across domains by introducing cross connections from one base network to another and vice versa, letting them benefit from each other. These ideas are sharpened into the proposed collaborative cross networks (CoNet). CoNet is achieved in simple multi-layer feedforward networks by using dual shortcut connections and joint loss functions, which can be trained efficiently by back-propagation.

The paper is organized as follows. We firstly introduce the preliminaries in Section 2, including notations and the base network. In Section 3, we then present an intuitive model to realize the cross-domain recommendation and point out several intrinsic weaknesses which limit its use. We propose a novel deep transfer learning approach for cross-domain recommendation, named collaborative cross networks (CoNet) in Section 4. The core component is the cross connection units which enable knowledge transfer between source and target networks (Sec. 4.1). Its adaptive variant enforces the sparse structure which adaptively controls when to transfer (Sec. 4.3). In Section 5, we experimentally show the benefits of both transfer learning and deep learning for improving the recommendation performance in terms of ranking metrics (Sec. 5.2). We show the necessity of adaptively selecting representations to transfer (Sec. 5.3). We reduce tens of thousands training examples without performance degradation by comparing with non-transfer models (Sec. 5.4), which can be used to save the cost/labor of labelling data. We review related works in Section 6 and conclude the paper in Section 7.

2 PRELIMINARY

We first give the notations and describe the problem setting (Sec. 2.1). We then review a multi-layer feedforward neural network as the base network for collaborative filtering (Sec. 2.2).

2.1 Notation

We are given two domains, a source domain $\mathcal{S}$ (e.g., news recommendation) and a target domain $\mathcal{T}$ (e.g., app recommendation). As a running example, we let app recommendation be the target domain and news recommendation be the source domain. The set of users in both domains are shared, denoted by $\mathcal{U}$ (of size $m = |\mathcal{U}|$). Denote the set of items in $\mathcal{S}$ and $\mathcal{T}$ by $\mathcal{I}_S$ and $\mathcal{I}_T$ (of size $n_S = |\mathcal{I}_S|$ and $n_T = |\mathcal{I}_T|$), respectively. Each domain is a problem of collaborative filtering for implicit feedback [17, 31]. For the target domain, let a binary matrix $\boldsymbol{R}_T \in \mathbb{R}^{m \times n_T}$ describe user-app installing interactions, where an entry $r_{ui} \in \{0, 1\}$ is 1 (observed entries) if user u has an interaction with app i and 0 (unobserved) otherwise. Similarly, for the source domain, let another binary matrix $\boldsymbol{R}_S \in \mathbb{R}^{m \times n_S}$ describe user-news reading interactions, where the entry $r_{uj} \in \{0, 1\}$ is 1 if user u has an interaction with news j and 0 otherwise. Usually the interaction matrix is very sparse since a user only consumed a very small subset of all items.

For the task of item recommendation, each user is only interested in identifying top-N items. The items are ranked by their predicted scores:

$$\hat{r}_{ui} = f(u, i|\Theta), \tag{1}$$

where f is the interaction function and Θ are model parameters. For matrix factorization (MF) techniques, the match function is the fixed dot product:

$$\hat{r}_{ui} = \boldsymbol{P}_u^T \boldsymbol{Q}_i, \tag{2}$$

and parameters are latent vectors of users and items $\Theta = \{\boldsymbol{P}, \boldsymbol{Q}\}$ where $\boldsymbol{P} \in \mathbb{R}^{m \times d}, \boldsymbol{Q} \in \mathbb{R}^{n \times d}$ and d is the dimension. For neural CF approaches, neural networks are used to parameterize function f and learn it from interactions:

$$f(\boldsymbol{x}_{ui}|\boldsymbol{P}, \boldsymbol{Q}, \theta_f) = \phi_o(\phi_L(...(\phi_1(\boldsymbol{x}_{ui}))...)), \tag{3}$$

where the input $\boldsymbol{x}_{ui} = [\boldsymbol{P}^T \boldsymbol{x}_u, \boldsymbol{Q}^T \boldsymbol{x}_i]$ is merged from projections of the user and the item, and the projections are based on their one-hot encodings $\boldsymbol{x}_u \in \{0, 1\}^m, \boldsymbol{x}_i \in \{0, 1\}^n$ and embedding matrices $\boldsymbol{P} \in \mathbb{R}^{m \times d}, \boldsymbol{Q} \in \mathbb{R}^{n \times d}$. The output and hidden layers are computed by ϕ_o and $\{\phi_l\}$ in a multilayer feedforward neural network (FFNN), and the connection weight matrices and biases are denoted by θ_f.

In our transfer/multitask learning approach for cross-domain recommendation, each domain is modelled by a neural network and these networks are jointly learned to improve the performance through mutual knowledge transfer. We review the base network in the following subsection before introducing the proposed model.

2.2 Base Network

We adopt an FFNN as the base network to parameterize the interaction function (see Eq.(3)). The base network is similar to the Deep model in [4, 6] and the MLP model in [12]. The base network, as shown in Figure 2 (the gray part or the blue part), consists of four modules with the information flow from the input (u, i) to the output $\hat{r}_{ui}$ as follows.

Input : $(u, i) \rightarrow \boldsymbol{x}_u, \boldsymbol{x}_i$. This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i, and maps them into one-hot encodings $\boldsymbol{x}_u \in \{0, 1\}^m$ and $\boldsymbol{x}_i \in \{0, 1\}^n$ where only the element corresponding to that index is 1 and all others are 0.

Embedding : $\boldsymbol{x}_u, \boldsymbol{x}_i \rightarrow \boldsymbol{x}_{ui}$. This module embeds one-hot encodings into continuous representations via two embedding matrices and then merges them as $\boldsymbol{x}_{ui} = [\boldsymbol{P}^T \boldsymbol{x}_u, \boldsymbol{Q}^T \boldsymbol{x}_i]$ to be the input of successive hidden layers.

Hidden layers: $\boldsymbol{x}_{ui} \rightsquigarrow \boldsymbol{z}_{ui}$. This module takes the continuous representations from the embedding module and then transforms, through multi-hop say L, to a final latent representation $\boldsymbol{z}_{ui} = \phi_L(...(\phi_1(\boldsymbol{x}_{ui})...)$. This module consists of multiple hidden layers to learn nonlinear interaction between users and items.

Output : $\boldsymbol{z}_{ui} \rightarrow \hat{r}_{ui}$. This module predicts the score $\hat{r}_{ui}$ for the given user-item pair based on the representation $\boldsymbol{z}_{ui}$ from the last layer of multi-hop module. Since we focus on one-class collaborative filtering, the output is the probability that the input pair is a positive interaction. This can be achieved by a softmax layer:

$$\hat{r}_{ui} = \phi_o(\boldsymbol{z}_{ui}) = \frac{1}{1 + \exp(-\boldsymbol{h}^T \boldsymbol{z}_{ui})}, \tag{4}$$

where $\boldsymbol{h}$ is the parameter.

3 CROSS-STITCH NETWORKS

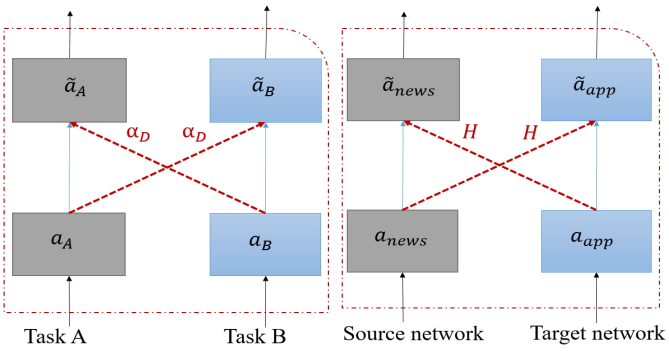
We first introduce an intuitive model to realize cross-domain recommendation using neural networks, and point out several intrinsic strong assumptions limiting its use, which inspire the design of our model in the next section.

Given two activation maps a_A and a_B from the l-th layer for two tasks A and B, cross-stitch convolutional networks (CSN) [27] learn linear combinations $\tilde{a}_A, \tilde{a}_B$ of both the input activations and feed these combinations as input to the successive layers' filters (see Fig. 1a):

$$\tilde{a}_A^{ij} = \alpha_S a_A^{ij} + \alpha_D a_B^{ij}, \quad \tilde{a}_B^{ij} = \alpha_S a_B^{ij} + \alpha_D a_A^{ij}, \tag{5}$$

where the shared parameter α_D controls information shared/transferred from the other network, α_S controls information from the task-specific network, and (i, j) is the location in the activation map.

Although the cross-stitch unit indeed incorporates knowledge from the source domain (and target domain vice versa), there are several limitations of this simple stitch unit. Firstly, cross-stitch networks cannot process the case that the dimensions of contiguous layers are different. In other words, it assumes that the activations in successive layers are in the *same vector space*. This is not an issue in convolutional networks for computer vision since the activation maps of contiguous layers are in the same space [21]. For collaborative filtering, however, it is not the case in typical multi-layer FFNNs where the architecture follows a tower pattern: the lower layers are wider and higher layers have smaller number of neurons [6, 12]. Secondly, it assumes that the representations from other networks are *equally important* with weights being all the same scalar α_D. Some features, however, are more useful and predictive and it should be learned attentively from data [43]. Thirdly, it assumes that the representations from other networks are *all useful* since it transfers activations from every location in a dense way. The sparse structure, however, plays a key role in general learning paradigm [8].



(a) Cross-stitch unit **(b) Cross connection unit**

Figure 1: The cross-stitch unit [27] (left) and the proposed cross connection unit (right). To enable knowledge transfer, the shared α_D in the cross-stitch network is a scalar while the shared H in the proposed collaborative cross network is a matrix.

Instead, our model can be extended to learn the sparse structure on the task relationship matrices which are defined in Eq. (9), with the help of the existing sparsity-induced regularization. As we will see in the experiments (see Table 2 and Figure 3), the sparse structure is necessary for generalization performance.

4 COLLABORATIVE CROSS NETWORKS

To alleviate the limitations of the cross-stitch networks, we propose collaborative cross networks (CoNet) to transfer knowledge for the cross-domain recommendation. The core component is cross connection units (Sec. 4.1). Our cross unit generalizes the cross-stitch (Sec. 4.2) and exploits the sparse structure (Sec. 4.3). We describe the model learning from implicit feedback datasets and the optimization process of the joint loss (Sec. 4.4). A complexity analysis is also given (Sec. 4.5).

4.1 Cross Connections Unit

In this section, we present a novel soft-sharing approach for transferring knowledge for cross-domain recommendation. It relaxes the hard-sharing assumption [47] and is motivated by the cross-stitch networks [27].

We now introduce the cross connections unit to enable dual knowledge transfer as shown in Fig. 1b. The central idea is simple, using a matrix rather than a scalar to transfer. Similarly to the cross-stitch network, the target network receives information from the source network and vice versa. In detail, let $\boldsymbol{a}_{app}$ be the representations of the l-th hidden layer and $\tilde{\boldsymbol{a}}_{app}$ be the input to the $l+1$-th in the app network, respectively. Similarly, they are $\boldsymbol{a}_{news}$ and $\tilde{\boldsymbol{a}}_{news}$ in the news network. The cross unit implements as follows:

$$\tilde{\boldsymbol{a}}_{app} = \boldsymbol{W}_{app}\boldsymbol{a}_{app} + \boldsymbol{H}\boldsymbol{a}_{news}, \tag{6a}$$

$$\tilde{\boldsymbol{a}}_{news} = \boldsymbol{W}_{news}\boldsymbol{a}_{news} + \boldsymbol{H}\boldsymbol{a}_{app}, \tag{6b}$$

where $\boldsymbol{W}_{app}$ and $\boldsymbol{W}_{news}$ are weight matrices, and the matrix $\boldsymbol{H}$ controls the information from news network to app network and vice versa. The knowledge transferring happens in two directions, from source to target and from target to source. We enable dual knowledge transfer across domains and let them benefit from each other. When target domain data is sparse, the target network can still learn a good representation from that of the source network through the cross connection units. It only needs to learn "residual" target representations with the reference of source representations, making the target task learning easier and hence alleviating the data sparse issues. The role of matrix $\boldsymbol{H}$ is similar to the scalar α_D in the sense of enabling knowledge transfer between domains.

We give a closer look at the matrix $\boldsymbol{H}$ for it can alleviate all three issues faced by cross-stitch unit. Firstly, the successive layers can be in *different vector space* (spaces with different dimensions) since the matrix $\boldsymbol{H}$ can be used to match their dimension. For example, if the l-th layer ($\boldsymbol{a}_{app}$ and $\boldsymbol{a}_{news}$) has dimension 128, and the $l+1$-th layer ($\tilde{\boldsymbol{a}}_{app}$ and $\tilde{\boldsymbol{a}}_{news}$) has dimension 64, then the matrix $\boldsymbol{H} \in \mathbb{R}^{64\times 128}$. Secondly, the entries of $\boldsymbol{H}$ are learned from data. They are likely not to be all the same, showing that the *importances* of transferred representations are different for each neuron/position. Thirdly, we can enforce some prior on the matrix $\boldsymbol{H}$ to exploit the structure of the neural architecture. The sparse structure can be enforced to adaptively *select useful representations* to transfer. Based on the cross connection units, we propose the CoNet models in the following sections, including a basic model (Sec. 4.2) and an adaptive variant (Sec. 4.3).

4.2 Basic Model

We propose the collaborative cross network (CoNet) model by adding cross connection units (see Sec. 4.1) and joint loss (see Eq.(12)) to the entire FFNN, as shown in Figure 2. We firstly describe a basic model in this section and then present an adaptive variant in the next section.

We decompose the model parameters into two parts, task-shared and task-specific:

$$\Theta_{app} = \{\boldsymbol{P}, (\boldsymbol{H}^l)_1^L\} \cup \{\boldsymbol{Q}_{app}, \theta_{f_{app}}\} \tag{7a}$$

$$\Theta_{news} = \{\boldsymbol{P}, (\boldsymbol{H}^l)_1^L\} \cup \{\boldsymbol{Q}_{news}, \theta_{f_{news}}\}, \tag{7b}$$

where $\boldsymbol{P}$ is the user embedding matrix and $\boldsymbol{Q}$ are the item embedding matrices with the subscript specifying the corresponding domain. The $\theta_f = \{(\boldsymbol{W}^l, b^l)_{l=1}^L, \boldsymbol{h}\}$ are the connection weight matrices and biases in the L-layer FFNN where $\boldsymbol{h}$ is the output weight as shown in Eq.(4). We stack the cross connections units on the top of the shared user embeddings, enabling deep knowledge transfer. Denote by $\boldsymbol{W}^l$ the weight matrix connecting from the l-th to the $l+1$-th layer (we ignore biases for simplicity), and by $\boldsymbol{H}^l$ the linear projection underlying the corresponding cross connections. Then two base networks are coupled by cross connections:

$$\boldsymbol{a}_{app}^{l+1} = \sigma(\boldsymbol{W}_{app}^l \boldsymbol{a}_{app}^l + \boldsymbol{H}^l \boldsymbol{a}_{news}^l), \tag{8a}$$

$$\boldsymbol{a}_{news}^{l+1} = \sigma(\boldsymbol{W}_{news}^l \boldsymbol{a}_{news}^l + \boldsymbol{H}^l \boldsymbol{a}_{app}^l), \tag{8b}$$

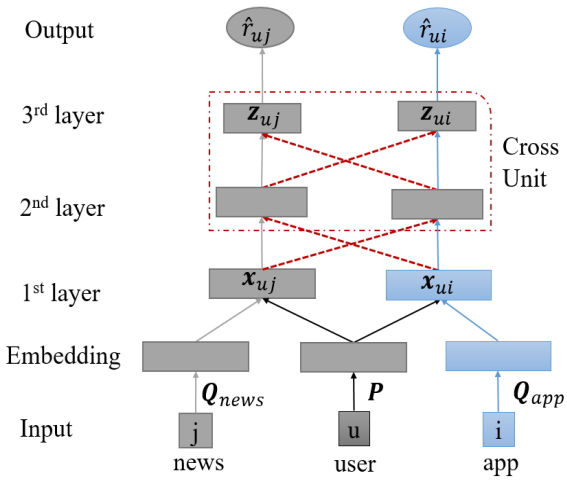


Figure 2: The proposed collaborative cross networks (CoNet) architecture (a version of three hidden layers and two cross units). We adopt a multilayer FFNN as the base network (grey or blue part, see Sec. 2.2). The red dotted lines indicate the cross connections which enable the dual knowledge transfer across domains. A cross unit is illustrated in the dotted rectangle box (see Fig. 1b).

where the function $\sigma(\cdot)$ is the widely used rectified activation units (ReLU) [29]. We can see that $\boldsymbol{a}_{app}^{l+1}$ receives two information flows: one is from the *transform* gate controlled by $\boldsymbol{W}_{app}^l$ and one is from the *transfer* gate controlled by $\boldsymbol{H}^l$ (similarly for the $\boldsymbol{a}_{news}^{l+1}$ in source network). We call $\boldsymbol{H}^l$ the relationship/transfer matrix since it learns to control how much sharing is needed. To reduce model parameters and make the model compact, we use the same linear transformation $\boldsymbol{H}^l$ for two directions, similar to the cross-stitch networks. Actually, using different matrices for two directions did not improve results on the evaluated datasets.

4.3 Adaptive Model

As we can see, the task relationship matrices $\{\boldsymbol{H}^l\}$ are crucial to the proposed CoNet model. We further enforce these matrices to have some structure. The assumption is that not all representations from another network are useful. We may expect that the representations coming from other domains are sparse and selective. This corresponds to enforcing a sparse prior on the structure and can be achieved by penalizing the task relationship matrix $\{\boldsymbol{H}^l\}$ via some regularization. It may help the individual network to learn intrinsic representations for itself and other tasks. In other words, $\{\boldsymbol{H}^l\}$ adaptively controls when to transfer.

We adopt the widely used sparsity-induced regularization—least absolute shrinkage and selection operator (lasso) [39]. In detail, denote by $r \times p$ the size of matrix $\boldsymbol{H}^l$ (usually $r = p/2$). That is, $\boldsymbol{H}^l$ linearly transforms representations $\boldsymbol{a}_{news}^l \in \mathbb{R}^p$ in the news network and the result is as part of

the input to the next layer $\tilde{\boldsymbol{a}}_{app}^{l+1} \in \mathbb{R}^r$ in the app network (see Eq.(8) and Eq.(6)). Denote by h_{ij} the (i,j) entry of $\boldsymbol{H}^l$. To induce overall sparsity, we impose the ℓ_1-norm penalty on the entries $\{h_{ij}\}$ of $\boldsymbol{H}^l$:

$$\Omega(\boldsymbol{H}^l) = \lambda \sum\nolimits_{i=1}^{r} \sum\nolimits_{j=1}^{p} |h_{ij}|, \tag{9}$$

where hyperparameter λ controls the degree of sparsity. This corresponds to the lasso regularization. We call this sparse variant as the SCoNet model.

Other priors like low-rank ($\boldsymbol{H} \approx \boldsymbol{U}^T\boldsymbol{V}$) factorization are alternatives of sparse structure. And the lasso variants like group lasso and sparse group lasso are also possible. We adopt the general sparse prior and the widely used lasso regularization. The others are left for future work.

4.4 Model Learning

Due to the nature of the implicit feedback and the task of item recommendation, the squared loss $(\hat{r}_{ui} - r_{ui})^2$ is not suitable since it is usually for rating regression/prediction. Instead, we adopt the cross-entropy loss:

$$\mathcal{L}_0 = -\sum\nolimits_{(u,i)\in \boldsymbol{R}^+ \cup \boldsymbol{R}^-} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log(1 - \hat{r}_{ui}), \tag{10}$$

where $\boldsymbol{R}^+$ and $\boldsymbol{R}^-$ are the observed interaction matrix and randomly sampled negative examples [31], respectively. This objective function has probabilistic interpretation and is the negative logarithm likelihood of the following likelihood function:

$$L(\Theta|\boldsymbol{R}^+ \cup \boldsymbol{R}^-) = \prod\nolimits_{(u,i)\in \boldsymbol{R}^+} \hat{r}_{ui} \prod\nolimits_{(u,i)\in \boldsymbol{R}^-} (1 - \hat{r}_{ui}), \tag{11}$$

where Θ are model parameters.

Now we define the joint loss function, leading to the proposed CoNet model which can be trained efficiently by back-propagation. Instantiating the *base loss* ($\mathcal{L}_0$) described in Eq. (10) by the *loss of app* ($\mathcal{L}_{app}$) and *loss of news* ($\mathcal{L}_{news}$) recommendation, the objective function for the CoNet model is their joint losses:

$$\mathcal{L}(\Theta) = \mathcal{L}_{app}(\Theta_{app}) + \mathcal{L}_{news}(\Theta_{news}), \tag{12}$$

where model parameters $\Theta = \Theta_{app} \cup \Theta_{news}$. Note that Θ_{app} and Θ_{news} share user embeddings and transfer matrices $\{\boldsymbol{P}, (\boldsymbol{H}^{\boldsymbol{l}})_{l=1}^{L}\}$. For the CoNet-sparse model, the objective function is added by the term $\Omega(\boldsymbol{H}^l)$ in Eq.(9).

The objective function can be optimized by stochastic gradient descent (SGD) and its variants like adaptive moment method (Adam) [19]. The update equations are:

$$\Theta^{new} \leftarrow \Theta^{old} - \eta \frac{\partial L(\Theta)}{\partial \Theta}, \tag{13}$$

where η is the learning rate. Typical deep learning library like TensorFlow[1] provides automatic differentiation and hence we omit the gradient equations $\frac{\partial L(\Theta)}{\partial \Theta}$ which can be computed by chain rule in back-propagation (BP).

[1]https://www.tensorflow.org

Table 1: Datasets and Statistics.

Dataset	#Users	Target Domain			Source Domain		
		#Items	#Interactions	Density	#Items	#Interactions	Density
Mobile	23,111	14,348	1,164,394	0.351%	29,921	617,146	0.089%
Amazon	80,763	93,799	1,323,101	0.017%	35,896	963,373	0.033%

4.5 Complexity Analysis

The model parameters Θ include $\{\boldsymbol{P}, (\boldsymbol{H}^{\boldsymbol{l}})_{l=1}^{L}\} \cup \{\boldsymbol{Q}_{app}, (\boldsymbol{W}_{app}^l, b_{app}^l)_{l=1}^{L}, \boldsymbol{h}_{app}\} \cup \{\boldsymbol{Q}_{news}, (\boldsymbol{W}_{news}^l, b_{news}^l)_{l=1}^{L}, \boldsymbol{h}_{news}\}$, where the embedding matrices $\boldsymbol{P}$, $\boldsymbol{Q}_{app}$ and $\boldsymbol{Q}_{news}$ contain a large number of parameters since they depend on the input size of users and items. Typically, the number of neurons in a hidden layer is about one hundred. That is, the size of connection weight matrices and task relationship matrices is hundreds by hundreds. In total, the size of model parameters is linear with the input size and is close to the size of typical latent factors models [20] and neural CF approaches [12].

During training, we update the target network using the target domain data and update the source network using the source domain data. The learning procedure is similar to the cross-stitch networks [27]. And the cost of learning each base network is approximately equal to that of running a typical neural CF approach [12]. In total, the entire network can be efficiently trained by BP using mini-batch stochastic optimization.

5 EXPERIMENT

We conduct thorough experiments to evaluate the proposed models. We show their superior performance over the state-of-the-art recommendation algorithms in a wide range of baselines (Sec. 5.2) and demonstrate the effectiveness of the sparse variant to select representations (Sec. 5.3). We quantify the benefit of knowledge transfer by reducing training examples (Sec. 5.4). Furthermore, we conduct investigations on the sensitivity to hyperparameter (Sec. 5.5). We analyze the optimization efficiency (Sec. 5.6) to help understand the proposed models.

5.1 Experimental Setup

We begin the experiments by introducing the datasets, evaluation protocol, baselines, and implementation details.

Dataset We evaluate on two real-world cross-domain datasets. The first dataset, **Mobile**[2], is provided by a large internet company, i.e., Cheetah Mobile[3]. The information contains logs of user reading news, the history of app installation, and some metadata such as news publisher and user gender collected in one month in the US. The dataset we used contains 1,164,394 user-app installations and 617,146 user-news reading records. There are 23,111 shared users, 14,348 apps, and 29,921 news articles. We aim to improve the app recommendation by transferring knowledge from relevant news reading domain. The data sparsity is over 99.6%.

[2]An anonymous version can be released later.
[3]http://www.cmcm.com/en-us/

The second dataset is a public **Amazon** dataset[4], which has been widely used to evaluate the performance of collaborative filtering approaches [11]. We use the two largest categories, Books and Movies & TV, as the cross-domain. We convert the ratings of 4-5 as positive samples. The dataset we used contains 1,323,101 user-book ratings and 963,373 user-movie ratings. There are 80,763 shared users, 93,799 books, and 35,896 movies. We aim to improve the book recommendation by transferring knowledge from relevant movie watching domain. The data sparsity is over 99.9%. The statistics are summarized in Table 1. As we can see, both datasets are very sparse and hence we hope improve performance by transferring knowledge from auxiliary domains.

Evaluation Protocol For item recommendation task, the leave-one-out (LOO) evaluation is widely used and we follow the protocol in [12]. That is, we reserve one interaction as the test item for each user. We determine hyper-parameters by randomly sampling another interaction per user as the validation/development set. We follow the common strategy which randomly samples 99 (negative) items that are not interacted by the user and then evaluate how well the recommender can rank the test item against these negative ones.

Since we aim at top-N item recommendation, the typical evaluation metrics are hit ratio (HR), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR), where the ranked list is cut off at $topN = 10$. HR intuitively measures whether the reserved test item is present on the top-N list, defined as:

$$HR = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \delta(p_u \leq topN),$$

where p_u is the hit position for the test item of user u, and $\delta(\cdot)$ is the indicator function. NDCG and MRR also account for the rank of the hit position, respectively defined as:

$$NDCG = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \frac{\log 2}{\log(p_u + 1)}, \; MRR = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \frac{1}{p_u}.$$

A higher value indicates better performance.

Baseline We compare with various baselines:

Baselines	Shallow method	Deep method
Single-domain	BPRMF [36]	MLP [12]
Cross-domain	CDCF [25], CMF [38]	MLP++, CSN [27]

BPRMF: Bayesian personalized ranking [36] is a typical latent factors CF approach which learns the user and item factors via matrix factorization and pairwise rank loss. It computes the prediction score by $\hat{r}_{ui} = \boldsymbol{P}_u^T \boldsymbol{Q}_i$ (see Eq.(2)). It is a shallow model and learns on the target domain only.

MLP: Multilayer perceptron [12] is a typical neural CF approach which learns user-item interaction function using neural networks. MLP corresponds to the base network as described in Section 2.2. It is a deep model and learns on the target domain only.

MLP++: We combine two MLPs by sharing the user embedding matrix only. This is a degenerated CoNet which has no cross connection units. It is a simple/shallow knowledge transfer approach applied to two domains.

CDCF: Cross-Domain CF with factorization machines (FM) [25] is a state-of-the-art cross-domain recommendation which extends FM [35]. It is a context-aware approach which applies factorization on the merged domains (aligned by the shared users). That is, the auxiliary domain is used as context. On the Mobile dataset, the context for a user in the target app domain is her history of reading news in the source news domain. Similarly, the context for a user in the target book domain is her history of watching movies in the source movie domain on the Amazon dataset. The feature vector for the input is a sparse vector $\boldsymbol{x} \in \mathbb{R}^{m+n_T+n_S}$ where the non-zero entries are as follows: 1) the index for user id, 2) the index for item id (target domain), and all indices for her reading articles/watching movies (source domain). It showed better performance than other cross-domain methods like triadic (tensor) factorization [16]. It is a shallow cross-domain model.

CMF: Collective matrix factorization [38] is a multi-relation learning approach which jointly factorizes matrices of individual domains. Here, the relation is user-item interaction. On Mobile, the two matrices are $\boldsymbol{A}$ = "user by app" and $\boldsymbol{B}$ = "user by news" respectively. Similarly, they are $\boldsymbol{A}$ = "user by book" and $\boldsymbol{B}$ = "user by movie" on Amazon. The shared user factors $\boldsymbol{P}$ enable knowledge transfer between two domains. Then CMF factorizes matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ simultaneously by sharing the user latent factors: $\boldsymbol{A} \approx \boldsymbol{P}^T \boldsymbol{Q}_A$ and $\boldsymbol{B} \approx \boldsymbol{P}^T \boldsymbol{Q}_B$. It is a shallow model and jointly learns on two domains. This can be thought of a non-deep transfer/multitask learning approach for cross-domain recommendation.

CSN: The cross-stitch network method [27], described in Section 3, is a good competitor. It is a deep multitask learning model which jointly learns two base networks. It enables knowledge transfer via a linear combination of activation maps from two networks via a shared coefficient, i.e., α_D in Eq.(5). This is a deep transfer/multitask learning approach for cross-domain recommendation.

Implementation For BPRMF, we use LightFM's implementation[5] which is a popular CF library. For CDCF, we adapt the official libFM implementation[6]. For MLP, we use the code released by its authors[7]. For CMF, we use a Python version reference to the original Matlab code[8]. Our methods are implemented using TensorFlow. Parameters are randomly initialized from Gaussian $\mathcal{N}(0, 0.01^2)$. The optimizer is Adam with initial learning rate 0.001. The size of mini batch is 128. The ratio of negative sampling is 1. As for the design of network structure, we adopt a tower pattern, halving the layer size for each successive higher layer. Specifically, the configuration of hidden layers in the base network is $[64 \rightarrow 32 \rightarrow 16 \rightarrow 8]$. This is also the network configuration of the MLP model. For CSN, it requires that the number of neurons in each hidden layer is the same. The configuration notation $[64] * 4$ equals $[64 \rightarrow 64 \rightarrow 64 \rightarrow 64]$. We investigate several typical configurations.

[4] http://snap.stanford.edu/data/web-Amazon.html

[5] https://github.com/lyst/lightfm

[6] http://www.libfm.org

[7] https://github.com/hexiangnan/neural_collaborative_filtering

[8] http://www.cs.cmu.edu/~ajit/cmf/

Table 2: Comparison results of different methods on two datasets. The best results are boldfaced and the best baselines are marked with stars.

Dataset	Metric	BPRMF	CMF	CDCF	MLP	MLP++	CSN	CoNet	SCoNet	improve	paired t-test
Mobile	HR	.6175	.7879	.7812	.8405	.8445	.8458*	.8480	**.8583**	1.47%	$p = 0.20$
	NDCG	.4891	.5740	.5875	.6615	.6683	.6733*	.6754	**.6887**	2.29%	$p = 0.25$
	MRR	.4489	.5067	.5265	.6210	.6268	.6366*	.6373	**.6475**	1.71%	$p = 0.34$
Amazon	HR	.4723	.3712	.3685	.5014	.5050*	.4962	.5167	**.5338**	5.70%	$p = 0.02$
	NDCG	.3016	.2378	.2307	.3143	.3175*	.3068	.3261	**.3424**	7.84%	$p = 0.03$
	MRR	.2971	.1966	.1884	.3113*	.3053	.2964	.3163	**.3351**	7.65%	$p = 0.05$

5.2 Comparing Different Approaches

In this section, we report the recommendation performance of different methods and discuss the findings. Table 2 shows the results of different models on the two datasets under three ranking metrics. The last two columns are the relative improvement and its paired t-test of our model vs. the best baselines. We can see that our proposed neural models are better than the base network (MLP), the shallow cross-domain models (CMF and CDCF) learned using two domains information, and the deep cross-domain model (MLP++ and CSN) on both datasets.

On Mobile, our model achieves 4.28% improvements in terms of MRR comparing with the non-transfer MLP, showing the benefits of knowledge transfer. Note that, the way of pre-training an MLP on source domain and then transferring user embeddings to target domain as warm-up did not achieve much improvement. In fact, the improvement is so small that it can be ignored. It shows the necessity of dual knowledge transfer in a deep way. Our model improves more than 20% in terms of MRR comparing with CDCF and CMF, showing the effectiveness of deep neural approaches. Together, our neural models consistently give better performance than other existing methods. Within our models (SCoNet vs CoNet), enforcing sparse structure on the task relationship matrices are useful. Note that, the dropout technique and ℓ_2 norm penalty did not achieve these improvements. They may harm the performance in some cases. It shows the necessity of selecting representations.

On Amazon, our model achieves 7.84% improvements in terms of NDCG comparing with the best baselines (MLP++), showing the benefits of knowledge transfer. Compared to the BPRMF, the inferior performance of CMF and CDCF shows the difficulty in transferring knowledge between Amazon Books and Movies, but our models also achieve good results. Comparing MLP++ and MLP, sharing user embedding is sightly better than the base network due to shallow knowledge transfer. Within our models, enforcing sparse structure on the task relationship matrices are also useful.

CSN is inferior to the proposed CoNet models on both datasets. Moreover, it is surprising that the CSN has some difficulty in benefitting from knowledge transfer on the Amazon dataset since it is inferior to the non-transfer base network MLP. The reason is possibly that the assumptions of CSN are not appropriate: all representations from the auxiliary domain are *equally* important and are *all* useful. By using a matrix $\boldsymbol{H}$ rather than a scalar α_D, we can relax the first assumption. And by enforcing a sparse structure on the matrix, we also relax the second assumption.

Note that the relative improvement of the proposed model vs. the best baseline is more significant on the Amazon dataset than that on the Mobile dataset, though the Amazon is much sparser than the Mobile (see Table 1). One explanation is that the relatedness the book and movie domains is much larger than that between the app and news domains. This will benefit all cross-domain methods including CMF, CDCF, and CSN, since they exploit information from both two domains. Another possibility is that the noise from auxiliary domain proposes a challenge for transferring knowledge. This shows that the proposed model is more effective since it can select useful representations from the source network and ignore the noisy ones. In the next section, we give a closer look at the impact of the sparse structure.

5.3 Impact of Sparsity: Selecting Representations to Transfer

On two real-world datasets, it both shows the usefulness of enforcing sparse structure on the task relationship matrices $\boldsymbol{H}$. We now quantify the contributions of the sparsity to CoNet. We investigate the impact of the sparsity by controlling the difference of architectures between CSN and CoNet. That is, we let them have the same architecture configuration. As a consequence, the performance of ablation comes from different means of knowledge transfer: scalar α_D used in CSN and sparse matrix $\boldsymbol{H}$ used in SCoNet.

Figure 3 shows the results on the Mobile and Amazon datasets under several typical architectures. We can see that the sparsity contributes to performance improvements and it is necessary to introduce the sparsity in general settings. On the Mobile data, introducing the sparsity improves the NDCG by relatively 2.29%. On the Amazon data, introducing the sparsity improves the NDCG by relatively 4.21%. These results show that it is beneficial to introduce the sparsity and to select representations to transfer on both datasets.

5.4 Benefit of Transferring: Reducing Labelled Data

Transfer learning can reduce the labor and cost of labelling data instances. In this section, we quantify the benefit of knowledge transfer by comparing with non-transfer methods.

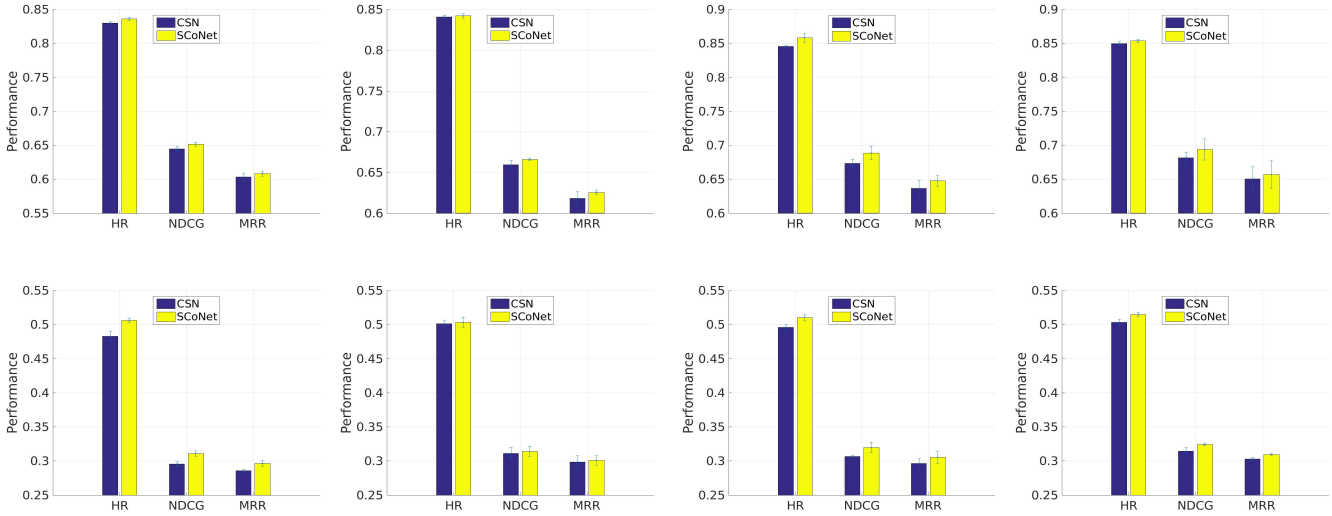


Figure 3: Impact of the sparsity on the Mobile (top) and Amazon (bottom) datasets. From left to right, configurations are $\{16, 32, 64, 80\} * 4$**.**

That is, we gradually reduce the number of training examples in the target domain until the performance of the proposed model is inferior to the non-transfer MLP model. The more training examples we can reduce, the more benefit we can get from transferring knowledge.

Referring to Table 1, there are about 50 examples per user on the Mobile dataset. We gradually reduce one and two training examples per user, respectively, to investigate the benefit of knowledge transfer. The results are shown in Table 3 where the rows corresponding to reduction percentage 0% are copied from Table 2 for clarity. The number 2.05% is approximately corresponding to reducing one training example per user. The results show that we can save the cost of labelling about $30,000$ training examples by transferring knowledge from the news domain but still have comparable performance with the MLP model, a non-transfer baseline.

According to Table 1, there are about 16 examples per user on the Amazon dataset. With a similar setting to the Mobile dataset, the results shown in Table 3 indicates that we can save the cost of labelling about $20,000$ training examples by transferring knowledge from movie domain. Note that the Amazon dataset is extremely sparse (the density is only 0.017%), implying that there is difficulty in acquiring many training examples. Under this scenario, our transfer models are an effective way of alleviating the issue of data sparsity and the cost of collecting data.

5.5 Sensitivity Analysis

We analyze the sensitivity to the sparse penalty which controls the sparsity (λ in Eq.(9)). Results are shown on the Mobile data only due to space limit and we give the corresponding conclusions on the Amazon data. Figure 4a shows

Table 3: The performance when reducing training examples. Results with stars are inferior to MLP.

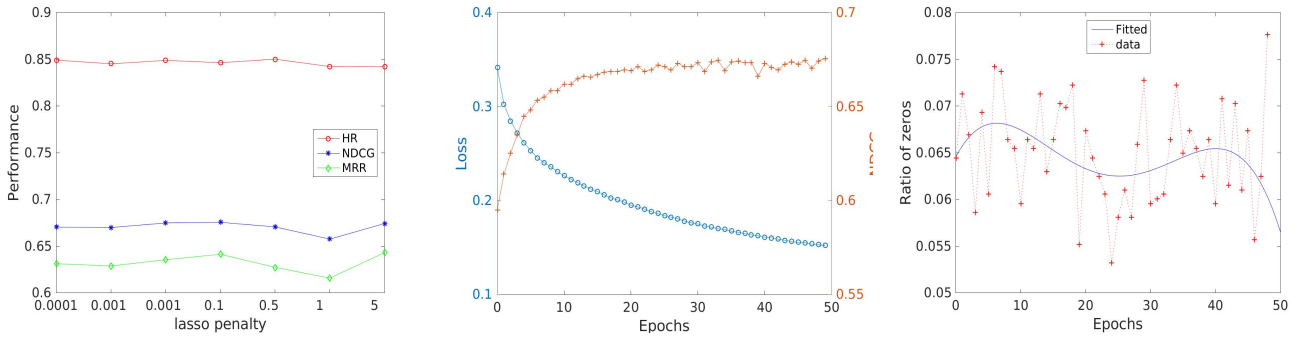
Dataset	Method	Reduction		HR	NDCG	MRR
		percent	amount			
Mobile	MLP	0%	0	.8405	.6615	.6210
	SCoNet	0%	0	.8547	.6802	.6431
		2.05%	23,031	.8439	.6640	.6238
		4.06%	45,468	.8347*	.6515*	.6115*
Amazon	MLP	0%	0	.5014	.3143	.3113
	SCoNet	0%	0	.5338	.3424	.3351
		1.11%	12,850	.5110	.3209	.3080*
		2.18%	25,318	.4946*	.3082*	.2968*

the performance varying with the penalty of sparsity enforcing on the task relationship matrices $\boldsymbol{H}$. On the Mobile data, the performance achieves good results at 0.1 (default) and 5.0, and it is 0.1 (default) and 1.0 on the Amazon data (not shown).

5.6 Optimization Performance

We analyze the optimization performance of SCoNet varying with training epochs. (One epoch means that the algorithm goes through the whole training dataset one time.) Results are shown on the Mobile dataset only due to space limit and the trend on the Amazon dataset is similar.

We firstly show the training loss and test performance. Figure 4b shows the training loss (averaged/normalized over all training examples) and NDCG test performance on the test set (HR and MRR have similar trends) varying with each optimization iteration. We can see that with more iterations, the training losses gradually decrease and the recommendation performance is improved accordingly. The most effective



(a) Sparse penalty **(b) Loss and performance** **(c) Ratio of zero entries**

Figure 4: Sensitivity and Optimization

updates are occurred in the first 15 iterations, and performance gradually improves until 30 iterations. With more iterations, SCoNet is relatively stable.

Since the sparsity of transfer matrices $(\boldsymbol{H}^l)_1^L$ is crucial to select representations for transferring, we show the change of zero entries over training epochs. For clarity and due to space limit, we only show the results of the first transfer matrix $\mathbf{H}^1$ which connects the first and the second hidden layers. Figure 4c shows the results where we use a 4-order polynomial to robustly fit the data. We can see that the matrix becomes sparser for the first 25 iterations, and the general trend is to sparsify. The average percent of zero entries in $\mathbf{H}^1$ is 6.5%. For the second and third transfer matrices, the percentage becomes 6.0% and 6.3%, respectively. In summary, sparse transfer matrices are learned and they can adaptively select partial representations to transfer across domains. And it may be better to *transfer many instead of all* representations at hand.

For the training time, our models spend about 100 seconds per epoch using one Nvidia TITAN Xp GPU. As a reference, it is 70s for MLP and 90s for CSN, which indicates that the training cost of the proposed method is comparable to (non-)transfer deep baselines.

6 RELATED WORKS

Recommender systems Recommender systems aim at learning user preferences on unknown items from their past history. Content-based recommendations are based on the matching between user profiles and item descriptions [34]. It is difficult to build the profile for each user when there is no/few content. Collaborative filtering (CF) alleviates this issue by predicting user preferences based on the user-item interaction behavior, agnostic to the content [7]. Latent factor models learn feature vectors for users and items mainly based on matrix factorization (MF) [20] which has probabilistic interpretations [24, 28]. MF is also flexible to integrate text [13, 18], social relations [15, 44], and implicit feedback [14, 20]. Factorization machines can mimic MF [35]. Some hierarchical methods can reduce to factorize a specific matrix [40]. Random walk and heterogeneous networks are adapted for recommendation [37, 42]. Neural networks are proposed to push the learning of feature vectors towards non-linear representations [6, 9, 12]. CF models, however, suffer from the data sparsity issue.

Cross-domain recommendation [2] is an effective technique to alleviate sparse issue. A class of methods are based on MF applied to each domain, including collective MF (CMF) [38] with its heterogeneous variants [33] and codebook transfer [22, 23]. Active learning [49] can construct entity correspondence with limited budget. Heterogeneous cross-domain [45] and multiple source domains [26] are also proposed to account for different cases of input. These are all shallow methods and have the difficulty in learning complex (highly nonlinear) user-item interaction relationship [6, 12, 41]. We follow this research thread by using deep networks to learn the nonlinear interaction function.

Transfer and multitask learning Transfer learning (TL) aims at improving the performance of the target domain by exploiting knowledge from source domains [32]. The typical TL technique in neural networks is two-step: initialize a target network with transferred features from a pre-trained source network [30, 47]. Different from this approach, we transfer knowledge in a deep way such that two base networks benefit from each other during the learning procedure. Similar to TL, the multitask learning (MTL) is to leverage useful knowledge in multiple related tasks to help each other [3, 48]. Multi-view learning [10] is closely related to MTL. The cross-stitch network (CSN) [27] enables information sharing between two base networks. We generalize CSN by relaxing the underlying assumption, especially via the idea of selecting representations to transfer.

7 CONCLUSIONS

We proposed a novel deep transfer learning for cross-domain recommendation. The sparse target user-item interaction matrix can be reconstructed with the knowledge guidance from the source domain, alleviating the data sparse issue. We demonstrated the necessity of adaptively selecting representations from the auxiliary domain to transfer. It may

harm the performance by transferring all of them with equal importance. We found that naive deep transfer models may be inferior to the shallow/neural non-transfer methods in some cases. Our transfer model can reduce tens of thousands training examples by comparing with the non-transfer methods without performance degradation. This is useful when collecting data is difficult or costly. Experiments demonstrate the effectiveness of the proposed models on two large real-world datasets by comparing with shallow/deep, single/cross-domain methods. As a future work, we will integrate content information into the collaborative cross network for alleviating the cold-start problem.

Acknowledgment We thank Cheetah Mobile for providing the Mobile dataset. The research has been supported by National Grant Fundamental Research (973 Program) of China under Project 2014CB340304, Hong Kong CERG projects 16211214/16209715/16244616, and NSFC 61673202. The work is also supported by HKPFS PF15-16701.

REFERENCES

[1] S. Berkovsky, T. Kuflik, and F. Ricci. 2007. Cross-domain mediation in collaborative filtering. In *UMAP*.
[2] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. 2015. Cross-domain recommender systems. In *Recommender Systems Handbook*.
[3] R. Caruana. 1997. Multitask Learning. *Machine Learning* (1997).
[4] H.-T. Cheng, L. Koc, J. Harmsen, et al. 2016. Wide & deep learning for recommender systems. In *ACM Recsys Workshop*.
[5] R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.
[6] P. Covington, J. Adams, and E. Sargin. 2016. Deep neural networks for youtube recommendations. In *ACM RecSys*.
[7] M. Deshpande and G. Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*.
[8] C. Doersch and A. Zisserman. 2017. Multi-Task Self-Supervised Visual Learning. In *IEEE CVPR*.
[9] G. Dziugaite and D. Roy. 2015. Neural network matrix factorization. *arXiv:1511.06443*.
[10] A. Elkahky, Y. Song, and X. He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*.
[11] R. He and J. McAuley. 2016. VBPR: visual Bayesian Personalized Ranking from implicit feedback. In *AAAI*.
[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. 2017. Neural collaborative filtering. In *WWW*.
[13] G.-N. Hu and X.-Y. Dai. 2017. Integrating reviews into personalized ranking for cold start recommendation. In *PAKDD*.
[14] G.-N. Hu, X.-Y. Dai, F.-Y. Qiu, R. Xia, T. Li, S.-J. Huang, and J.-J. Chen. 2018. Collaborative Filtering with Topic and Social Latent Factors Incorporating Implicit Feedback. *ACM Transactions on Knowledge Discovery from Data* (2018).
[15] G.-N. Hu, X.-Y. Dai, Y. Song, S.-J. Huang, and J.-J. Chen. 2015. A Synthetic Approach for Recommendation: Combining Ratings, Social Relations, and Reviews. In *IJCAI*.
[16] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. 2013. Personalized recommendation via cross-domain triadic factorization. In *WWW*.
[17] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *IEEE ICDM*.
[18] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *ACM RecSys*.
[19] D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.
[20] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009).
[21] A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.
[22] B. Li, Q. Yang, and X. Xue. 2009. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. In *IJCAI*.
[23] B. Li, X. Zhu, R. Li, C. Zhang, X. Xue, and X. Wu. 2011. Cross-domain collaborative filtering over time. In *IJCAI*.
[24] B. Liu, Y. Fu, Z. Yao, and H. Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *ACM SIGKDD*.
[25] B. Loni, Y. Shi, M. Larson, and A. Hanjalic. 2014. Cross-Domain Collaborative Filtering with Factorization Machines. In *ECIR*.
[26] Z. Lu, E. Zhong, L. Zhao, E. Xiang, W. Pan, and Q. Yang. 2013. Selective transfer learning for cross domain recommendation. In *SIAM International Conference on Data Mining*.
[27] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. 2016. Cross-stitch networks for multi-task learning. In *IEEE CVPR*.
[28] A. Mnih and R. Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*.
[29] V. Nair and G. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.
[30] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE CVPR*.
[31] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-class collaborative filtering. In *IEEE ICDM*.
[32] S. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* (2010).
[33] W. Pan, N. Liu, E. Xiang, and Q. Yang. 2011. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *IJCAI*.
[34] M. Pazzani and D. Billsus. 2007. Content-based recommendation systems. In *The adaptive web*.
[35] S. Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology* (2012).
[36] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.
[37] C. Shi, Z. Zhang, P. Luo, P. Yu, Y. Yue, and B. Wu. 2015. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *ACM CIKM*.
[38] A. Singh and G. Gordon. 2008. Relational learning via collective matrix factorization. In *ACM SIGKDD*.
[39] R. Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B* (1996).
[40] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *ACM SIGIR*.
[41] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu. 2016. Personal recommendation using deep recurrent neural networks in NetEase. In *IEEE ICDE*.
[42] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. 2010. Temporal recommendation on graphs via long-and short-term preference fusion. In *ACM SIGKDD*.
[43] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.
[44] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han. 2017. Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation. In *ACM SIGKDD*.
[45] D. Yang, J. He, H. Qin, Y. Xiao, and W. Wang. 2015. A graph-based recommendation across heterogeneous domains. In *ACM CIKM*.
[46] Z. Yang, R. Salakhutdinov, and W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *ICLR*.
[47] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. 2014. How transferable are features in deep neural networks?. In *NIPS*.
[48] Y. Zhang and Q. Yang. 2017. A survey on multi-task learning. *arXiv:1707.08114*.
[49] L. Zhao, S. Pan, E. Xiang, E. Zhong, Z. Lu, and Q. Yang. 2013. Active transfer learning for cross-system recommendation. In *AAAI*.