Cross-Domain Recommendation: An Embedding and Mapping Approach

Tong Man[1,2,*], **Huawei Shen**[1,2,†], **Xiaolong Jin**[1,2,†], and **Xueqi Cheng**[1,2,†]
[1]{CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences, China}
[2]{University of Chinese Academy of Sciences, China}
[*]supermt@gmail.com, [†]{shenhuawei, jinxiaolong, cxq}@ict.ac.cn

Abstract

Data sparsity is one of the most challenging problems for recommender systems. One promising solution to this problem is cross-domain recommendation, i.e., leveraging feedbacks or ratings from multiple domains to improve recommendation performance in a collective manner. In this paper, we propose an Embedding and Mapping framework for Cross-Domain Recommendation, called EMCDR. The proposed EMCDR framework distinguishes itself from existing cross-domain recommendation models in two aspects. First, a multi-layer perceptron is used to capture the nonlinear mapping function across domains, which offers high flexibility for learning domain-specific features of entities in each domain. Second, only the entities with sufficient data are used to learn the mapping function, guaranteeing its robustness to noise caused by data sparsity in single domain. Extensive experiments on two cross-domain recommendation scenarios demonstrate that EMCDR significantly outperforms state-of-the-art cross-domain recommendation methods.

1 Introduction

With the explosively growing amount of online information, recommender system becomes an important tool to help users efficiently find their desired information [Adomavicius and Tuzhilin, 2005; Shen *et al.*, 2014]. Collaborative filtering is widely adopted in recommender systems [Sarwar *et al.*, 2001; Koren *et al.*, 2009], which assumes that users and items that are similar in history will also be similar in future. Among existing collaborative filtering methods, Matrix Factorization (MF) gains great success in the last decade [Mnih and Salakhutdinov, 2007]. MF formalizes the recommendation problem into a matrix completion problem, recovering a user-item rating matrix by factorizing it into a low-rank user matrix and a low-rank item matrix. The recommendation accuracy of MF heavily depends on the rating matrix. Unfortunately, the rating matrix is often highly sparse in practical scenarios, forming a hard barrier for MF to be widely used in realistic recommender systems [Schein *et al.*, 2002; Lin *et al.*, 2013; Zhou *et al.*, 2011]. Moreover, the rating matrix is unevenly distributed, i.e., a majority of inactive users express preference on a small number of items and a majority of unpopular items get few feedbacks. Such a data sparsity problem is particularly severe for newly joining users and newly arriving items, forming the so-called cold-start problem.

Cross-domain recommendation was proposed to combat the long-standing data sparsity problem, which leverages feedbacks or ratings from multiple domains to improve recommendation accuracy in a collective manner [Singh and Gordon, 2008; Zhang *et al.*, 2012]. Existing methods for cross-domain recommendation fall into two main types, dealing with the data sparsity problem in different scenarios. The first type works in an *asymmetric* way, aiming to leverage data in an auxiliary domain to alleviate data sparsity of the target domain [Li *et al.*, 2009a; 2009b; Pan *et al.*, 2010]. In this scenario, knowledge or patterns learned from the auxiliary domain is directly transferred to the target domain, acting as regularization or priors to improve recommendation accuracy. For these methods, the key is to identify the appropriate knowledge that can be transferred from the auxiliary domain to the target one. However, without fully leveraging the data in both domains, the capacity of these methods is limited. The second type of methods works in a *symmetric* way, assuming that both the auxiliary and target domains suffer from the data sparsity problem, and anticipates that these domains can complement each other [Singh and Gordon, 2008; Agarwal *et al.*, 2011; Jamali and Lakshmanan, 2013; Li and Lin, 2014; Liu *et al.*, 2015]. In this scenario, there is no distinction between the auxiliary domain and the target domain, i.e., both domains are treated equally. All domains work in a collective way to combat data sparsity. These methods generally learn a mapping function across domains, explicitly distinguishing domain-specific factors from domain-sharing factors. The main drawback or concern about these methods is that learning both domain-specific and domain-sharing factors is likely to further exacerbate data sparsity.

In this paper, we study the cross-domain recommendation problem from an embedding and mapping perspective, which is in line with the aforementioned second type of methods, i.e., learning a mapping function across domains. We investigate two key issues of cross-domain recommendation: (1) How to represent the mapping function across domains, in linear or nonlinear format? (2) Which part of data should be leveraged to learn the mapping function, all available

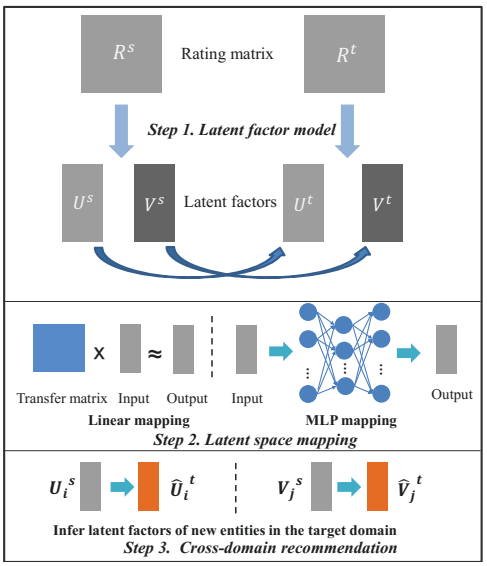

Figure 1: Illustrative diagram of the EMCDR framework.

data in both domains or just part of them? These two questions are very tricky: While a nonlinear mapping function has high flexibility in characterizing the complex relationship across domains, it requires more data for training and may be easily over-fitted in cross-domain recommendation with a high data sparsity problem. Meanwhile, not all the users or items are useful for learning the mapping function. It has been shown that the users/items that are inactive/unpopular in both domains is harmful in learning a cross-domain mapping function [Man *et al.*, 2015]. In this paper, we give our answers to the above two questions and thus propose an Embedding and Mapping framework for Cross-Domain Recommendation, called EMCDR. We validate the effectiveness of EMCDR on two cross-domain recommendation scenarios by comparing it to state-of-the-art cross-domain recommendation methods.

2 The Proposed EMCDR Framework

In this section, we formulate the cross-domain recommendation problem and present the EMCDR framework.

Suppose we have two domains which share the same users and/or items. Users/items appearing in only one domain can be regarded as the upcoming users/items in the other domain. In this sense, these two domains also share the same users and items. Without loss of generality, one domain is referred to as the source domain and the other as the target domain. Let $\mathcal{U} = \{u_1, u_2, ...\}$ and $\mathcal{V} = \{v_1, v_2, ...\}$ be the user and item sets, R^s and R^t be two rating matrices from the source and target domains respectively, where R^s_{ij} is the rating that user u_i gives to item v_j in the source domain and R^t_{ij} is the corresponding rating in the target domain.

Given two partially observed matrices, R^s and R^t, and cross-domain user and item sets, $\mathcal{U}$ and $\mathcal{V}$, we aim to make recommendation for those users and/or items in the target domain with little information, by leveraging information across domains. For this purpose, we propose the Embedding and Mapping framework for Cross-Domain Recommendation (EMCDR). This framework contains three major steps, i.e., latent factor modeling, latent space mapping and cross-domain recommendation, as illustrated in Figure 1.

Algorithm 1 The EMCDR framework.

Require:
Source domain R^s, target domain R^t;
User set $\mathcal{U}$, item set $\mathcal{V}$;
Ensure:
Make recommendation for entities in the target domain: $\{u \in R^s \& u \notin R^t\}$ / $\{v \in R^s \& v \notin R^t\}$;
Latent Factor Model
1: Learn $\{U^s, V^s\}$ from R^s;
2: Learn $\{U^t, V^t\}$ from R^t;
Latent Space Mapping
3: Learn the mapping function $f_U(\cdot)$ ($f_V(\cdot)$) by users across domains
Cross-domain Recommendation
4: Get affine factors $\hat{U}^t$ ($\hat{V}^t$) of target users
5: Make recommendation for target users (items)

At the first step, we aim to find the representation of users/items in the latent space by latent factor models [Koren *et al.*, 2009; Man *et al.*, 2016]. Latent factor model assumes that there is a factor associated with each user and each item, and the observed user-item ratings are actually the results of interactions between the latent factors of users and items. The factors of users and items in latent spaces can be represented as feature vectors. We denote the latent factors learned from the source domain R^s as $\{U^s, V^s\}$ and the target domain as $\{U^t, V^t\}$. At the second step, we model the cross-domain relationships of users/items through a mapping function. We assume that there is an underlying mapping relationship between the user/item feature vectors of the source and target domains, and further use a mapping function to capture this relationship. Formally, given user u_i's feature vectors in the two domains, U^s_i and U^t_i, we adopt a mapping function $f(\cdot;\theta)$ to capture the cross-domain relationship $U^t_i = f(U^s_i;\theta)$, where θ is the parameter of the mapping function. Finally, at the third step, we make recommendation for one new user/item in the target domain. We can obtain the corresponding latent factor in the target domain, with the help of the latent factor in the source domain and the mapping function. The complete EMCDR framework is presented in Algorithm 1.

3 The EMCDR Models

In this section, we implement the proposed EMCDR framework into different models with two different latent factor models, and two latent space mapping functions.

3.1 Latent Factor Modeling

As aforesaid, the first step of the EMCDR framework aims to learn the latent factors of entities (i.e., users and/or items) in both source and target domains, respectively. In this paper, two different models, namely MF [Koren *et al.*, 2009] and BPR [Rendle *et al.*, 2009], are employed in the framework,

offering us a rating-oriented recommendation algorithm and a ranking-oriented one. MF decomposes a rating matrix into two low-dimensional matrices. Let R be an $|\mathcal{U}| \times |\mathcal{V}|$ rating matrix, K be the dimension of latent factors, U be the $K \times |\mathcal{U}|$ user latent factor matrix where the ith column U_i represents the latent factor of user u_i, and V be the $K \times |\mathcal{V}|$ item latent factor matrix where the jth column V_j represents the latent factor of item v_j. With Gaussian observation noise, the probability that the rating R_{ij} is observed for one user-item pair (u_i, v_j) is modeled as [Mnih and Salakhutdinov, 2007]

$$p(R_{ij}|U_i, V_j; \sigma^2) = \mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2), \tag{1}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2.

MF maximizes the probability of the conditional distribution over observed rating matrix R, forming the following optimization problem

$$\min_{U,V} \left(\sum_i \sum_j ||I_{ij} \cdot (R_{ij} - U_i^T V_j)||_F^2 + \lambda_U \sum_i ||U_i||_F^2 + \lambda_V \sum_j ||V_j||_F^2 \right), \tag{2}$$

where the indicator variable $I_{ij} = 1$ if user i has a rating on item j; otherwise, $I_{ij} = 0$. $||\cdot||_F^2$ denotes the Frobenius norm, λ_U and λ_V are the coefficients for the regularization terms.

Different from MF that optimizes a rating based objective function, BPR adopts a preference ranking approach that optimizes a ranking based objective function. BPR first creates a training set $D : \mathcal{U} \times \mathcal{V} \times \mathcal{V}$, composed by preference pairs, from the original rating matrix

BPR矩阵分解

$$D := \{(u_i, v_j, v_l)|R_{ij} > R_{il}\}. \tag{3}$$

The probability of a preference pair (R_{ij}, R_{il}), given the latent factor $\{U_i, V_j, V_l\}$, is modeled as

$$p(R_{ij} > R_{il}) = \sigma(U_i^T V_j - U_i^T V_l), \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function. BPR then optimizes the following objective function, derived from the posterior probability of the latent factors

$$\min_{U,V} \left(\sum_{(u_i, v_j, v_l) \in D} -\ln \sigma(U_i^T V_j - U_i^T V_l) + \lambda_U \sum_i ||U_i||_F^2 + \lambda_V \sum_j ||V_j||_F^2 \right). \tag{5}$$

The parameters of both MF and BPR models can be estimated by optimizing the objective functions via stochastic gradient descent. The latent factors learned by the MF and BPR models (e.g., user factors U) can be viewed as a set of coordinates in the latent space. Later on, we employ these coordinates as implicit features to learn a mapping function across different domains. For the MF model, given a rating R_{ij} for the user-item pair (u_i, v_j), the updating rules are

$$\begin{aligned} U_i &\leftarrow U_i - \eta \cdot \{(R_{ij} - U_i^T V_j) V_j + \lambda_U U_i\}, \\ V_j &\leftarrow V_j - \eta \cdot \{(R_{ij} - U_i^T V_j) U_i + \lambda_V V_j\} \end{aligned} \tag{6}$$

where U_i and V_j represent the latent factors of user u_i and item v_j respectively, λ_U and λ_V are regularization terms, and η is the learning rate.

For the BPR model, a training set composed by preference pairs is first constructed. Then, in the learning step, given a preference pair $R_{ijl} = \{R_{ij} > R_{il}\}$, the updating rule is

$$\Theta \leftarrow \Theta + \eta \cdot \left(\frac{e^{-\hat{x}_{ijl}}}{1 + e^{-\hat{x}_{ijl}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{ijl} - \lambda_\Theta \Theta \right). \tag{7}$$

Here, $\Theta \in \{U_i, V_j, V_l\}$, $\hat{x}_{ijl} = U_i^T V_j - U_i^T V_l$, λ_Θ is regularization terms for latent factors, and η is the learning rate.

3.2 Latent Space Mapping

With the latent factor models, we can get the latent factors, $\{U^s, V^s, U^t, V^t\}$, of users and items in the source and target domains. We assume that the relationship across domains could be captured by a mapping function. In what follows, we propose two different mapping functions: one is a linear function, and the other is a nonlinear function based on Multi-Layer Perceptron (MLP) [Ruck *et al.*, 1990].

To obtain the mapping function, we formalize the learning procedure as a supervised regression problem. Specifically, we minimize the mapping loss to get the mapping function

$$\min_\theta \sum_{u_i \in \mathcal{U}} L(f_U(U_i^s; \theta), U_i^t), \tag{8}$$

where $L(\cdot, \cdot)$ is the loss function defined on the mapping result of the feature vector from the source domain and the corresponding feature vector from the target domain. Since the input and output of the mapping function are multi-dimensional numerical vectors, we choose square loss as the loss function.

Similarly, the optimization problem of the mapping function on the item side can be represented as

$$\min_\theta \sum_{v_j \in \mathcal{V}} L(f_V(V_j^s; \theta), V_j^t). \tag{9}$$

Linear Mapping. The latent space mapping problem can be solved using a linear mapping method [Mikolov *et al.*, 2013]. Here a mapping function is defined as a transfer matrix M. Taking the user side for example, the goal is to find M such that $M \times U_i^s$ approximates U_i^t for all users across source and target domains. In practice, M can be learned via the following optimization problem

$$\min_M \sum_{u_i \in \mathcal{U}} L(M \times U_i^s, U_i^t) + \Omega(M), \tag{10}$$

where $L(\cdot, \cdot)$ is the loss function and $\Omega(M)$ is the regularization on the transfer matrix.

MLP-based Nonlinear Mapping. In this study, we also employ an MLP to cope with the latent space matching problem, as shown in Figure 1. Adopting an MLP as the mapping function has the following advantages. First, as the input and output of the mapping function are K-dimensional vectors, MLP is convenient to capture such input-output structure. Second, MLP is a nonlinear transformation, which is more flexible

Table 1: Statistics of the MovieLens-Netflix dataset.

Statistics	MovieLens	Netflix
Num. of movies	5,871	5,871
Num. of users	69,258	70,132
Num. of ratings	7,891,832	11,658,783

Table 2: Statistics of the Douban dataset.

Statistics	Movie	Book
Num. of users	10,000	10,000
Num. of items	25,342	51,204
Num. of ratings	2,287,712	832,103

than a linear mapping function. Finally, MLP is easy to be optimized by back propagation methods [Le Cun *et al.*, 1990]. Specifically, the optimization problem can be formalized as

$$\min_{\theta} \sum_{u_i \in \mathcal{U}} L\big(f_{mlp}(U_i^s;\theta), U_i^t\big), \tag{11}$$

where $f_{mlp}(\cdot;\theta)$ is the MLP mapping function, θ is its parameter set, which are the weight matrices and bias terms between layers.

In order to obtain the MLP mapping function, in this paper we adopt stochastic gradient descent to learn the parameters. Looping through the training set, with any user u_i across both source and target domains, we refresh the parameters of the MLP. The back propagation algorithm is employed to calculate the gradients of the parameters. The iterative process is stopped until the model converges.

3.3 Cross-domain Recommendation

Given a user/item in the target domain, without sufficient information we cannot estimate a promising latent factor for it to make recommendation. However, with the latent factor learned for it in the source domain, as well as the mapping function between the source and target domains, we can get an affine latent factor for it in the target domain. For example, give a user u_i in the target domain, we can get its affine factor $\hat{U}_i^t$ by the following equation

$$\hat{U}_i^t = f(U_i^s;\theta), \tag{12}$$

where U_i^s is its latent factor in the source domain and $f(\cdot;\theta)$ is the mapping function. Next, the recommendation is done base on this affine latent factor.

4 Experiments

4.1 Experiments Setup

Datasets. Two real-world datasets are adopted for evaluation. The first dataset, MovieLens-Netflix, contains users and aligned movies from two public benchmark sets, namely the Netflix Prize and the MovieLens project. Over $5,000$ movies are shared across Netflix and MovieLens according to the IMDB information. Users in the two platforms are quite different, forming an item-shared cross-domain scenario. The second dataset was crawled from an online social network, i.e., Douban [Huang *et al.*, 2012], where users give rating to books and movies, forming a natural cross-domain scenario with shared users. Statistics of the two datasets are shown in Table 1 and Table 2. In this paper, we take MovieLens as source domain and Netflix as target domain for the item-shared scenario. For the user-shared scenario, movie is taken as source domain, and book is taken as target domain.

Experiment Setup. To evaluate the validity and efficiency of the proposed framework on the cross-domain recommendation task, we randomly remove all the rating information of a fraction of entities in the target domain and take them as cross-domain cold-start entities for making recommendation. For the sake of stringency of the experiments, we set different fractions for cold-start entities, namely, 10%, 20%, 30%, 40%, and 50%. In addition, since different sets of cold-start entities may affect the final recommendation results, we repeatedly sample users/items for L times to generate different sets. We report the average results and standard deviations over these L different sets. Specifically, we set L as 10 in the experiments.

Dimension K of the latent factor is set as $20, 50$, and 100. The learning rate and regularization coefficients are optimized via 5-fold cross validation on the training dataset. For the linear mapping function, the size of the permutation matrix is $K \times K$, and the regularization coefficient λ_M is chosen as 0.01; for the MLP mapping function, we choose the structure of the MLP as one-hidden layer, the dimension of the input and output of the MLP is set as K, whilst the number of nodes in the hidden layer is set as $2 \times K$. The weight and bias parameters of the MLP is initialized according to the rule in [Glorot and Bengio, 2010]. We use minibatch with a size of 16 and no momentum is used. Finally, a tan-sigmoid function is employed as the activation function.

Models for Comparison. In the experiments, we implement the EMCDR framework into four models by adopting MF or BPR as the latent factor models and with either a linear or an MLP based cross-domain mapping function:

- MF_EMCDR_LIN: It adopts MF as its latent factor model and a linear cross-domain mapping function;
- MF_EMCDR_MLP: It also adopts MF as its latent factor model, but has an MLP based cross-domain mapping function;
- BPR_EMCDR_LIN: It employs BPR as its latent factor model and a linear cross-domain mapping function;
- BPR_EMCDR_MLP: It also employs BPR as its latent factor model, but adopts an MLP based cross-domain mapping function.

We compare these EMCDR models with the following baseline models and algorithms for validating their merits:

- AVE: It predicts cross-domain entities with the global average ratings in the target domain;
- CMF [Singh and Gordon, 2008]: In CMF, the latent factors of entities are shared across the source and target domains; 把不同的user-item数据放在同一个矩阵里训
- CST [Pan *et al.*, 2010]: It transfers the latent factors learned in the source domain into the target domain to

Table 3: Recommendation performance in terms of RMSE on the MovieLens-Netflix dataset.

		10%	20%	30%	40%	50%
	AVE	1.0845 ± 0.0011	1.0834 ± 0.0009	1.0853 ± 0.0009	1.0864 ± 0.0008	1.0859 ± 0.0013
K=20	CMF	0.9251 ± 0.0006	0.9280 ± 0.0011	0.9313 ± 0.0004	0.9337 ± 0.0009	0.9359 ± 0.0007
	CST	0.9170 ± 0.0007	0.9205 ± 0.0009	0.9220 ± 0.0006	0.9265 ± 0.0011	0.9311 ± 0.0005
	LFM	0.9162 ± 0.0008	0.9198 ± 0.0011	0.9223 ± 0.0007	0.9259 ± 0.0005	0.9290 ± 0.0005
	MF_EMCDR_LIN	0.9160 ± 0.0006	0.9192 ± 0.0007	0.9220 ± 0.0005	0.9258 ± 0.0009	0.9289 ± 0.0007
	MF_EMCDR_MLP	$\mathbf{0.8988 \pm 0.0009}$*	$\mathbf{0.9018 \pm 0.0006}$*	$\mathbf{0.9041 \pm 0.0008}$*	$\mathbf{0.9078 \pm 0.0005}$*	$\mathbf{0.9092 \pm 0.0004}$*
K=50	CMF	0.9226 ± 0.0004	0.9258 ± 0.0007	0.9279 ± 0.0009	0.9301 ± 0.0007	0.9328 ± 0.0005
	CST	0.9163 ± 0.0007	0.9193 ± 0.0010	0.9224 ± 0.0003	0.9229 ± 0.0003	0.9263 ± 0.0005
	LFM	0.9152 ± 0.0008	0.9177 ± 0.0007	0.9199 ± 0.0009	0.9220 ± 0.0005	0.9253 ± 0.0007
	MF_EMCDR_LIN	0.9141 ± 0.0011	0.9163 ± 0.0009	0.9189 ± 0.0007	0.9210 ± 0.0004	0.9241 ± 0.0007
	MF_EMCDR_MLP	$\mathbf{0.8960 \pm 0.0009}$*	$\mathbf{0.8987 \pm 0.0005}$*	$\mathbf{0.9015 \pm 0.0005}$*	$\mathbf{0.9040 \pm 0.0006}$*	$\mathbf{0.9068 \pm 0.0007}$*
K=100	CMF	0.9201 ± 0.0004	0.9224 ± 0.0007	0.9245 ± 0.0007	0.9273 ± 0.0011	0.9297 ± 0.0006
	CST	0.9137 ± 0.0007	0.9163 ± 0.0003	0.9193 ± 0.0005	0.9234 ± 0.0007	0.9265 ± 0.0006
	LFM	0.9131 ± 0.0005	0.9154 ± 0.0007	0.9179 ± 0.0009	0.9201 ± 0.0011	0.9233 ± 0.0009
	MF_EMCDR_LIN	0.9119 ± 0.0005	0.9142 ± 0.0007	0.9169 ± 0.0005	0.9193 ± 0.0007	0.9218 ± 0.0009
	MF_EMCDR_MLP	$\mathbf{0.8939 \pm 0.0007}$*	$\mathbf{0.8962 \pm 0.0005}$*	$\mathbf{0.8991 \pm 0.0009}$*	$\mathbf{0.9010 \pm 0.0007}$*	$\mathbf{0.9036 \pm 0.0008}$*

Significantly outperforms LFM at the: * 0.01 level, paired t-test

Table 4: Recommendation performance in terms of AUC on the MovieLens-Netflix dataset.

		10%	20%	30%	40%	50%
K=20	CMF	0.6132 ± 0.0005	0.6110 ± 0.0007	0.6093 ± 0.0006	0.6071 ± 0.0008	0.6048 ± 0.0007
	CST	0.6351 ± 0.0004	0.6312 ± 0.0007	0.6295 ± 0.0007	0.6265 ± 0.0007	0.6233 ± 0.0006
	LFM	0.6421 ± 0.0007	0.6402 ± 0.0006	0.6378 ± 0.0009	0.6350 ± 0.0010	0.6325 ± 0.0004
	BPR_EMCDR_LIN	0.6730 ± 0.0008	0.6702 ± 0.0009	0.6681 ± 0.0004	0.6652 ± 0.0007	0.6631 ± 0.0005
	BPR_EMCDR_MLP	$\mathbf{0.6892 \pm 0.0007}$*	$\mathbf{0.6869 \pm 0.0008}$*	$\mathbf{0.6828 \pm 0.0004}$*	$\mathbf{0.6803 \pm 0.0002}$*	$\mathbf{0.6782 \pm 0.0009}$*
K=50	CMF	0.6153 ± 0.0009	0.6132 ± 0.0007	0.6118 ± 0.0010	0.6094 ± 0.0007	0.6072 ± 0.0008
	CST	0.6372 ± 0.0003	0.6342 ± 0.0010	0.6332 ± 0.0007	0.6284 ± 0.0005	0.6263 ± 0.0006
	LFM	0.6445 ± 0.0009	0.6424 ± 0.0007	0.6401 ± 0.00011	0.6378 ± 0.0014	0.6350 ± 0.0007
	BPR_EMCDR_LIN	0.6752 ± 0.0007	0.6724 ± 0.0010	0.6700 ± 0.0009	0.6681 ± 0.0005	0.6659 ± 0.0009
	BPR_EMCDR_MLP	$\mathbf{0.6919 \pm 0.0004}$*	$\mathbf{0.6892 \pm 0.0009}$*	$\mathbf{0.6860 \pm 0.0004}$*	$\mathbf{0.6835 \pm 0.0009}$*	$\mathbf{0.6808 \pm 0.0004}$*
K=100	CMF	0.6176 ± 0.0009	0.6154 ± 0.0010	0.6142 ± 0.0006	0.6110 ± 0.0008	0.6095 ± 0.0009
	CST	0.6396 ± 0.0007	0.6364 ± 0.0009	0.6340 ± 0.0005	0.6305 ± 0.0011	0.6272 ± 0.0006
	LFM	0.6470 ± 0.0008	0.6448 ± 0.0010	0.6426 ± 0.0006	0.6399 ± 0.0006	0.6378 ± 0.0007
	BPR_EMCDR_LIN	0.6778 ± 0.0004	0.6746 ± 0.0005	0.6722 ± 0.0009	0.6703 ± 0.0010	0.6684 ± 0.0007
	BPR_EMCDR_MLP	$\mathbf{0.6948 \pm 0.0004}$*	$\mathbf{0.6922 \pm 0.0008}$*	$\mathbf{0.6900 \pm 0.0007}$*	$\mathbf{0.6876 \pm 0.0006}$*	$\mathbf{0.6852 \pm 0.0005}$*

Significantly outperforms LFM at the: * 0.01 level, paired t-test

combat the sparsity problem and improve the performance of cross-domain recommendation. This method is one state-of-the-art transfer learning method for cross-domain recommendation.

- LFM [Agarwal *et al.*, 2011]: In LFM, each entity has a global latent factor shared across domains. The domain-dependent factor for each entity is generated from the global one by multiplying it with a domain-specific transfer matrix. This method is one state-of-the-art method for cross-domain recommendation in symmetric way.

Several other cross-domain recommendation methods exist, with comparable performance with the selected baselines. Limited by space, we cannot list all of them in this paper. We select the above methods as baselines, since they are representative methods of two types of cross-domain recommendation, asymmetric and symmetric manners.

Metrics. For MF-based EMCDR models (MF_EMCDR_LIN and MF_EMCDR_MLP), we evaluate the recommendation performance in terms of Root Mean Squared Error (RMSE). For BPR-based EMCDR models (BPR_EMCDR_LIN and BPR_EMCDR_MLP), average AUC statistic is employed for evaluating the recommendation performance.

4.2 Experimental Results

Recommendation Performance. Experimental results of RMSE and AUC on the MovieLens-Netflix dataset are presented in Tables 3 and 4, respectively. Note that since the AVE recommendation method provides an equal recommendation for all user-item pairs, we do not report the AUC values for it. From these two tables, we can see that the AVE recommendation method exhibits the worst performance as a trivial method. The EMCDR models with either the linear or the nonlinear cross-domain mapping functions outperform all baseline models in terms of both RMSE and AUC metrics. Moreover, the results demonstrate that the EMCDR models with MLP mapping function bring significant improvement as compared to the EMCDR models with linear mapping function and other baselines. For example, when $K = 20$ and the target fraction of users is 10%, the average RMSE of the MF_EMCDR_MLP model is 0.8988, indicating that it outperforms the CMF model by 2.8%, the CST model by 2.0%, the LFM model by 1.9%, and the MF_EMCDR_LIN model by 1.9%. The average AUC of the BPR_EMCDR_MLP model is 0.6892, which outperforms the CMF, CST, LFM, and BPR_EMCDR_LIN models by 12.4%, 8.5%, 7.3%, and 2.4%, respectively.

Next, we report the experimental results on the Douban dataset, where users are shared across the user-movie and user-book domains. In this experiment, we randomly remove a fraction of users in the target domain and then make recommendation for these users. The experimental results in terms of RMSE and AUC on the Douban dataset are presented in Tables 5 and 6, respectively. It can be seen that the performance of the models on the Douban dataset is basically consistent with that on the MovieLens-Netflix dataset. For example, when $K = 20$ and the fraction of removed users is 10%, the average RMSE of the MF_EMCDR_MLP model is 0.7650, which performs better than the CMF model by 3.2%, the CST model by 2.6%, the LFM model by 1.9%, and the MF_EMCDR_LIN model by 0.7%. The aver-

Table 5: Recommendation performance in terms of RMSE on the Douban dataset.

		10%	20%	30%	40%	50%
	AVE	0.8360 ± 0.0005	0.8357 ± 0.0007	0.8346 ± 0.0011	0.8354 ± 0.0019	0.8339 ± 0.0024
K=20	CMF	0.7903 ± 0.0009	0.7932 ± 0.0007	0.7964 ± 0.0008	0.7992 ± 0.0009	0.8023 ± 0.0007
	CST	0.7851 ± 0.0006	0.7891 ± 0.0003	0.7911 ± 0.0007	0.7932 ± 0.0006	0.7951 ± 0.0004
	LFM	0.7801 ± 0.0011	0.7823 ± 0.0010	0.7849 ± 0.0009	0.7876 ± 0.0007	0.7902 ± 0.0011
	MF_EMCDR_LIN	0.7704 ± 0.0007	0.7725 ± 0.0006	0.7748 ± 0.0005	0.7781 ± 0.0007	0.7798 ± 0.0007
	MF_EMCDR_MLP	**0.7650 ± 0.0006***	**0.7672 ± 0.0005***	**0.7697 ± 0.0004***	**0.7718 ± 0.0010***	**0.7746 ± 0.0009***
K=50	CMF	0.7872 ± 0.0008	0.7893 ± 0.0005	0.7923 ± 0.0009	0.7957 ± 0.0004	0.7991 ± 0.0008
	CST	0.7810 ± 0.0007	0.7829 ± 0.0007	0.7856 ± 0.0010	0.7882 ± 0.0007	0.7917 ± 0.0007
	LFM	0.7774 ± 0.0011	0.7795 ± 0.0008	0.7819 ± 0.0008	0.7835 ± 0.0004	0.7862 ± 0.0007
	MF_EMCDR_LIN	0.7688 ± 0.0010	0.7702 ± 0.0006	0.7735 ± 0.0009	0.7751 ± 0.0005	0.7770 ± 0.0010
	MF_EMCDR_MLP	**0.7613 ± 0.0007***	**0.7632 ± 0.0006***	**0.7660 ± 0.0003***	**0.7682 ± 0.0008***	**0.7715 ± 0.0008***
K=100	CMF	0.7844 ± 0.0005	0.7873 ± 0.0007	0.7892 ± 0.0009	0.7930 ± 0.0004	0.7964 ± 0.0003
	CST	0.7782 ± 0.0006	0.7803 ± 0.0004	0.7823 ± 0.0011	0.7861 ± 0.0007	0.7897 ± 0.0005
	LFM	0.7727 ± 0.0007	0.7753 ± 0.0006	0.7785 ± 0.0009	0.7806 ± 0.0010	0.7830 ± 0.0010
	MF_EMCDR_LIN	0.7656 ± 0.0009	0.7683 ± 0.0008	0.7702 ± 0.0007	0.7723 ± 0.0008	0.7741 ± 0.0009
	MF_EMCDR_MLP	**0.7571 ± 0.0009***	**0.7594 ± 0.0009***	**0.7603 ± 0.0007***	**0.7641 ± 0.0008***	**0.7688 ± 0.0007***

Significantly outperforms LFM at the: * 0.01 level, paired t-test

Table 6: Recommendation performance in terms of AUC on the Douban dataset.

		10%	20%	30%	40%	50%
K=20	CMF	0.6553 ± 0.0007	0.6521 ± 0.0004	0.6502 ± 0.0007	0.6479 ± 0.0010	0.6451 ± 0.0009
	CST	0.6692 ± 0.0007	0.6682 ± 0.0006	0.6770 ± 0.0005	0.6746 ± 0.0009	0.6701 ± 0.0007
	LFM	0.6737 ± 0.0010	0.6712 ± 0.0009	0.6791 ± 0.0010	0.6772 ± 0.0006	0.6748 ± 0.0008
	BPR_EMCDR_LIN	0.7345 ± 0.0007	0.7310 ± 0.0004	0.7283 ± 0.0011	0.7251 ± 0.0008	0.7219 ± 0.0007
	BPR_EMCDR_MLP	**0.7593 ± 0.0005***	**0.7571 ± 0.0007***	**0.7551 ± 0.0009***	**0.7526 ± 0.0005***	**0.7498 ± 0.0008***
K=50	CMF	0.6574 ± 0.0008	0.6542 ± 0.0007	0.6525 ± 0.0011	0.6502 ± 0.0009	0.6475 ± 0.0012
	CST	0.6694 ± 0.0007	0.6673 ± 0.0009	0.6749 ± 0.0008	0.6735 ± 0.0007	0.6714 ± 0.0006
	LFM	0.6752 ± 0.0011	0.6734 ± 0.0008	0.6812 ± 0.0007	0.6797 ± 0.0008	0.6763 ± 0.0010
	BPR_EMCDR_LIN	0.7377 ± 0.0009	0.7336 ± 0.0012	0.7301 ± 0.0013	0.7276 ± 0.0007	0.7240 ± 0.0008
	BPR_EMCDR_MLP	**0.7612 ± 0.0004***	**0.7594 ± 0.0006***	**0.7579 ± 0.0009***	**0.7547 ± 0.0005***	**0.7516 ± 0.0007***
K=100	CMF	0.6597 ± 0.0004	0.6568 ± 0.0005	0.6527 ± 0.0009	0.6521 ± 0.0007	0.6496 ± 0.0008
	CST	0.6721 ± 0.0005	0.6523 ± 0.0008	0.6491 ± 0.0010	0.6452 ± 0.0004	0.6401 ± 0.0003
	LFM	0.6778 ± 0.0007	0.6589 ± 0.0011	0.6549 ± 0.0009	0.6545 ± 0.0007	0.6518 ± 0.0003
	BPR_EMCDR_LIN	0.7390 ± 0.0010	0.7354 ± 0.0010	0.7328 ± 0.0008	0.7297 ± 0.0006	0.7263 ± 0.0003
	BPR_EMCDR_MLP	**0.7633 ± 0.0008***	**0.7623 ± 0.0005***	**0.7593 ± 0.0007***	**0.7571 ± 0.0009***	**0.7534 ± 0.0003***

Significantly outperforms LFM at the: * 0.01 level, paired t-test

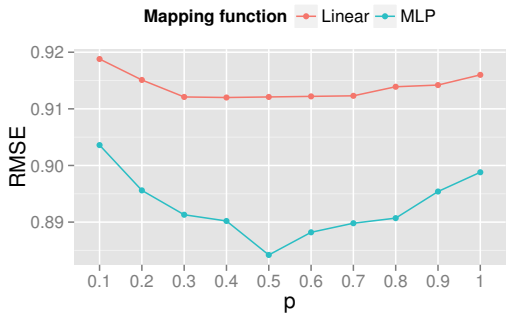

Figure 2: Performance of cross-domain recommendation with respect to different levels of training sets for $k = 20$ and the training size is 10%.

age AUC of the BPR_EMCDR_MLP model is 0.7593, suggesting performance improvement over the CMF, CST, LFM, and BPR_EMCDR_LIN models by 12.7%, 13.5%, 15.9%, and 3.4%, respectively.

Mapping Function Learning. We now investigate how the performance of EMCDR is affected by the entities that are used for learning the mapping function across domains. Taking cross-domain movie recommendation as an example, we choose the top-p percent of popular movies, with p ranging from 0.1 to 1. Results are shown in Figure 2. When p is small, the performance increases with p, indicating that the performance could be improved integrating more entities for training the mapping function. However, when p surpasses a threshold, the performance decreases with p. This indicates that when unpopular and even cold entities are involved in the learning of mapping function, and the performance of the cross-domain recommendation is negatively affected.

5 Conclusions

In this paper we presented an embedding-and-mapping framework for tackling cross-domain recommendation, called EMCDR. We first employed latent factor models to project users/items in both source and target domains into two different latent spaces. We then learned a mapping function between latent spaces to capture the coordinate relationship across the two domains. For a user/item in the target domain, we make recommendation by mapping its factor in the source domain into the target domain. Moreover, to learn an appropriate mapping function, we proposed a training set selection strategy based on the popularity of entities, where only active users and popular items are leveraged to learning the mapping function. Experiments conducted on two cross-domain recommendation scenarios convincingly demonstrate that the proposed models can significantly improve the quality of cross-domain recommendation.

Acknowledgments

This work was funded by the National Basic Research Program of China under grant numbers 2013CB329606 and 2014CB340401, and the National Natural Science Foundation of China under grant numbers 61472400, 61425016, 61572473, and 61433014. Huawei Shen is also funded by Youth Innovation Promotion Association CAS.

References

[Adomavicius and Tuzhilin, 2005] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[Agarwal *et al.*, 2011] Deepak Agarwal, Bee-Chung Chen, and Bo Long. Localized factor models for multi-context recommendation. In *SIGKDD '11*, 2011.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS '10*, 2010.

[Huang *et al.*, 2012] Junming Huang, Xue-Qi Cheng, Hua-Wei Shen, Tao Zhou, and Xiaolong Jin. Exploring social influence via posterior effect of word-of-mouth recommendations. In *WSDM '12*, 2012.

[Jamali and Lakshmanan, 2013] Mohsen Jamali and Laks Lakshmanan. Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In *WWW '13*, 2013.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Le Cun *et al.*, 1990] B Boser Le Cun, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS '90*, 1990.

[Li and Lin, 2014] Chung-Yi Li and Shou-De Lin. Matching users and items across domains to improve the recommendation quality. In *SIGKDD '14*, 2014.

[Li *et al.*, 2009a] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI '09*, 2009.

[Li *et al.*, 2009b] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *ICML '09*, 2009.

[Lin *et al.*, 2013] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *SIGIR '13*, 2013.

[Liu *et al.*, 2015] Yan-Fu Liu, CS NTHU, Cheng-Yu Hsu, Shan-Hung Wu, and NTHU EDU. Non-linear cross-domain collaborative filtering via hyper-structure transfer. In *ICML '15*, 2015.

[Man *et al.*, 2015] Tong Man, Hua-Wei Shen, Junming Huang, and Xue-Qi Cheng. Context-adaptive matrix factorization for multi-context recommendation. In *CIKM '15*, 2015.

[Man *et al.*, 2016] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. Predict anchor links across social networks via an embedding approach. In *IJCAI '16*, 2016.

[Mikolov *et al.*, 2013] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[Mnih and Salakhutdinov, 2007] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *NIPS '07*, 2007.

[Pan *et al.*, 2010] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI '10*, 2010.

[Pan *et al.*, 2011] Weike Pan, Nathan N Liu, Evan W Xiang, and Qiang Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *IJCAI '11*, 2011.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI '09*, 2009.

[Ruck *et al.*, 1990] Dennis W Ruck, Steven K Rogers, Matthew Kabrisky, Mark E Oxley, and Bruce W Suter. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *Neural Networks, IEEE Transactions on*, 1(4):296–298, 1990.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW '01*, 2001.

[Schein *et al.*, 2002] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02*, 2002.

[Shen *et al.*, 2014] Huawei Shen, Dashun Wang, Chaoming Song, and Albert-László Barabási. Modeling and predicting popularity dynamics via reinforced poisson processes. In *AAAI '14*, 2014.

[Singh and Gordon, 2008] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *SIGKDD '08*, 2008.

[Zhang *et al.*, 2012] Yu Zhang, Bin Cao, and Dit-Yan Yeung. Multi-domain collaborative filtering. *arXiv preprint arXiv:1203.3535*, 2012.

[Zhou *et al.*, 2011] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR '11*, 2011.