MetaEM: Meta Embedding Mapping for Federated Cross-domain Recommendation to Cold-Start Users

Dongyi Zheng[1], Yeting Guo[1], Fang Liu[2(✉)], Nong Xiao[1], and Lu Gao[3]

[1] College of Computer, National University of Defense Technology, Changsha, China
[2] School of Design, HuNan University, Changsha, China
fangl@hnu.edu.cn
[3] School of History, Hubei University, Wuhan, China

Abstract. Cross-domain recommendation exploits the rich data from source domain to solve the cold-start problem of target domain. Considering the recommendation system contains some user private information, how to provide accurate suggestions for cold-start users on the basis of protecting privacy is an important issue. Federated recommendation systems keep user private data on mobile devices to protect user privacy. However, compared to federated single-domain recommendation, federated cross-domain recommendation needs to train more models, making resource-constrained mobile devices infeasible to run large-scale models. In view of this, we design a meta embedding mapping method for federated cross-domain recommendation called MetaEM. The training stage of MetaEM includes pretraining and mapping. The pretrain stage learns user and item embeddings of source domain and target domain respectively. Items embeddings are divided into common and private. The common embeddings are shared by all users, and we train a meta-network to generate private embeddings for each user. The mapping stage learns to transfer user embeddings from source domain to target domain. In order to alleviate the negative impact of users with low number of ratings on mapping model, we employ a task-oriented optimization method. We implement the MetaEM prototype on large real-world datasets and extensive experiments demonstrate that MetaEM achieves the best performance and is more compatible with complicated models compared to other state-of-the-art baselines.

Keywords: Meta learning · Cross-domain recommendation · Federated learning · Cold-start · Embedding mapping

1 Introduction

Recommender systems have played an important role in various online applications of the Internet, which help users discover interesting content from massive

Supported by organization nudt.

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 154–172, 2022.
https://doi.org/10.1007/978-3-031-24383-7_9

information. Since the newly registered users of the system have not yet generated interaction data, it is difficult for the recommendation system to provide accurate recommendations for them, which is called the cold-start problem.

Cross-domain recommendation (CDR) provides a solution to the cold-start problem by utilizing the rich data of the source domain to improve the recommendation accuracy of the target domain. For example, recommend books to users based on their movie ratings. At present, cross-domain recommendation models can be divided into mapping-based [13], shared entity representation [5], heterogeneous graph embedding [19] and multi-domain collaborative training [17]. Among them, the mapping-based is the most used method, and the representative models of this method include EMCDR [13], TMCDR [20], PTUPCDR [22] and so on. However, these models all belong to the centralized training method, and dedicated to training high-performance cross-domain models. There is a risk of user data leakage in centralized training process.

Federated learning is a distributed machine learning framework that can be applied in recommendation systems to solve privacy protection issues. It saves users' private data on mobile devices, and then builds and trains recommendation models between mobile devices and cloud server. Existing federated recommendation models include FCF [1], FedRec [14], FedRec++ [8], FedFast [15], MetaMF [10], etc. However, these methods focus on addressing the privacy protection issue in single-domain recommendation scenarios. There is a lack of relevant research on privacy protection in cross-domain recommendation scenarios. Therefore, it is of great significance to study federated cross-domain recommendation, which can provide more accurate recommendation suggestions for cold-start users on the basis of protecting privacy. We mainly study federated cross-domain recommendation that belong to mapping-based.

We summarize the challenges faced by federated cross-domain recommendation as follows:

(1) Resource-constrained mobile devices. Compared with federated single-domain recommendation, map-based federated cross-domain recommendation requires federated training of three models: two pretrained models (including source and target domains) and a mapping model. Therefore, federated cross-domain recommendation requires more models to be trained on mobile devices than federated single-domain recommendation. Large-scale model training is difficult for resource-constrained mobile devices. In addition, since the mobile device needs to continuously download and upload the embedding and gradient information of the items in the pretrain stage, when the number of items is massive, the amount of data transmitted between mobile devices and cloud server increases sharply, which demands more RAM, and communication bandwidth on mobile phones. Therefore, it is not practical to allow resource-constrained mobile devices to participate in training. However, in the real world, most devices are resource constrained.
(2) Optimization Strategy. Existing cross-domain models such as [13,18] adopt a distance-oriented approach to optimize the mapping model, taking the Euclidean distance between the transformed embedding (prediction) and the target embedding (ground truth) as the optimization objective. However,

low-quality embeddings of users with fewer ratings can add negative effects to the mapping model. And each training sample consists of the user's target embedding and transformation embedding, and the number of total training samples depends on the number of training users, so the total number of samples is limited.

In response to the challenge of resource-constrained mobile devices, our goal is to design a light federated cross-domain recommendation model, which can significantly reduce the amount of data transferred between mobile devices and cloud servers, alleviating the communication and storage load of mobile devices. Facing the challenge of optimization strategy, we can use the transformed embedding for the prediction task. Our main contributions are summarized as follows:

- We design a meta embedding mapping method for federated cross-domain recommendation, MetaEM. It utilizes a meta-learner to generate small-scale private item embedding matrix for each user, thereby reducing the amount of data transferred between mobile devices and cloud servers.
- To stably learn the mapping model, we use a task-oriented optimization strategy to remove the side effects of low-quality user embeddings.
- We conduct extensive experiments to demonstrate the effectiveness and compatibility of MetaEM for cold-start users. It demonstrate that MetaEM achieves the best performance and is more compatible with complicated models compared to other state-of-the-art baselines.

2 Related Work

In this section, we discuss the existing related works, mainly including the application of federated learning in single-domain recommendation scenario and cross-domain recommendation models.

There are many applications of federated learning in single-domain recommendation scenarios. FedFast [15] puts forward an accelerated strategy of federated learning for recommendation. Because the traditional federated learning algorithm converges slowly for recommendation, it will continue to occupy the equipment resources of the client during model training. FedFast not only protects users' privacy and improves convergence speed, but also maintains fine recommendation performance. On the basis of user clustering, FedFast proposed two algorithms, user sampling and gradient aggregation, which can converge with fewer rounds. MetaMF [10] considers that traditional federated learning assumes that the server model is the same size as the client model, but not all users have high-quality mobile phones, so the client can't perform large-scale neural network training like the central server. Therefore, considering the resource limitations of clients, this paper proposes a federated recommendation algorithm based on meta-learning, aiming at learning a small personalized model for each client, so as to achieve accurate recommendation service. FedRec and FedRec++ [8,9] add randomly sampled items and simulated ratings to users, which makes the server unable to accurately identify users' preferences for items, thus improving the

protection of users' privacy. These methods are the latest applications of federated learning in single-domain recommendation scenarios. When traditional federated learning schemes adapt to single-domain recommendation, it will face various problems, such as slow convergence speed, large model size of client and privacy leakage from gradient. In the cross-domain recommendation scenario, these problems will be more serious.

The goal of cross-domain recommendation is to enrich the knowledge of target domain with the data of source domain. Tong et al. proposed a single-target cross-domain recommendation algorithm based on domain mapping (EMCDR) [13] and [7]. The recommendation goal of EMCDR is for users or items with little information in the target domain. For example, for users or items newly added to the system, their latent factors can not be found in the target domain, but the users are active in the source domain and have accurate latent factors in the source domain. So EMCDR map the latent factors in the source domain to the target domain through the mapping function, and then complete the recommendation. CMF [16] was proposed to realize cross-domain knowledge integration by connecting multiple rating matrices and sharing user latent factor across domains. Yong et al. proposed TMCDR [21], which divided cross-domain recommendation into Transfer stage, Meta stage and prediction stage. The Transfer stage is similar to the stage of learning user and item latent factors in EMCDR, then the trained model is used as a pretraining model. The Meta stage draws on the idea of meta-learning, extracts some user data from the user set in the cross domain as tasks, and repeatedly extracts several tasks to form a batch for training. In the prediction stage, the cold start user can be recommended by using the trained meta-network to output the latent factor in the target domain. Yong et al. also proposes a novel framework PTUPCDR [22]. It uses a meta-learner to model a personalized preference bridge with user traits extracted from the user's interaction history. Then use the personalized bridge to complete the mapping of the source domain embeddings to the target domain. All these existing CDR methods adopt a centralized architecture and are dedicated to training high-performance cross-domain recommendation models, ignoring that the data collection stage is also included in the real recommendation scenario. Therefore, user privacy is at risk of being leaked [11,12].

Unlike federated single-domain recommendation, which is dedicated to enhancing privacy protection, cross-domain recommendation faces more complex challenges while solving privacy protection, such as more consumption of mobile resources, and we devote to study an light and compatible federated learning paradigm for cross-domain recommendation scenarios. Compared with centralized cross-domain recommendation dedicated to training high-performance cross-domain models, we hope to improve the privacy protection of cross-domain recommendation by studying the federated cross-domain recommendation procedure and get competitive recommendation performance, while enabling real-world resource-constrained mobile devices to train stably.

3 MODEL

3.1 Overview

In cross-domain recommendation, there are a source domain and a target domain. The source domain has user set $\mathcal{U}^s$, item set $\mathcal{V}^s$, and rich rating matrix R^s. The target domain has user set $\mathcal{U}^t$, item set $\mathcal{V}^t$, and sparse rating matrix R^t, and cold-start user set $\mathcal{U}^c$, where the cold-start users have ratings in source domain and no ratings in target domain. We define the overlapping users $\mathcal{U}^o = \mathcal{U}^s \cap \mathcal{U}^t$, where $\mathcal{U}^c \in \mathcal{U}^o$, and $\mathcal{V}^s$ and $\mathcal{V}^t$ are non-overlapping. Important notations are shown in Table 1.

Table 1. Important notations

Symbol	Definition
$*^s$ and $*^t$	The notations of source domain and target domain
k_c	Dimension of shared item embedding
k_p	Dimension of private item embedding
m, n	Number of users and items
D_i	Local data for u_i
$\mathcal{U}^o$	Overlapping user sets for source and target domains
$\mathcal{U}^c$	Cold-start user set of target domain
$\mathcal{U}^{train}$	Train users in overlapping users
$V^c \in \mathbb{R}^{n \times k_c}$	Common item embedding matrix
$V^p \in \mathbb{R}^{k_c \times k_p}$	Private item embedding matrix
U^*, V^*	User and item embedding matrix
$V^*_{u_i}$	Private item embedding matrix for user i
$Server\_pt_{paras}$	Server-side parameters in pretrain stage
$Server\_mp_{paras}$	Server-side parameters in mapping stage
$Device_{paras}$	Device-side parameters
u_i, v_j	Embedding of user i and item j
$R \in \mathbb{R}^{m \times n}$	User-item rating matrix
$r_{ij} \in R$	The rating of user u_i on item v_j
$\hat{r}_{ij}$	The predicted rating of user u_i on item v_j
g_ϕ	Meta-network
f_θ	Mapping-network

Our goal is to accurately recommend items for $\mathcal{U}^c$. Our approach is to model the relationship between the embeddings of $\mathcal{U}^o$ in the source and target domains, and then use the trained model to transfer the embeddings of $\mathcal{U}^c$ from the source domain to the target domain and finally complete the recommendation.

3.2 Build MetaEM

Before introducing the design of each module of MetaEM in detail, we first introduce how to use MetaEM to build a federated recommendation system

between distributed mobile devices and cloud servers. The overview of MetaEM is shown in Fig. 1.

In aspect of module deployment, MetaEM is divided into MF module, meta-network module, mapping module and prediction module. In particular, we deploy the meta-network module, mapping module and MF module on the server side. The prediction module is deployed on mobile devices.

In the view of training stage. The modules and processes included in each stage are as follows:

(1) Pretrain stage. Both the source domain and the target domain need pretrain stage. The pretrain stage includes the MF module, the Meta module and the Prediction module. Server-side trainable parameters include: user embedding matrix, meta network, and MLP_{user}, called $Server\_pt_{paras}$. The trainable parameters on the mobile device side is MLP_{item}, called $Device_{paras}$. Taking the source domain as an example, the pretrain process of the target domain is similar to that of the source domain. In the training procedure, the server first initializes $Server\_pt_{paras}$, and then each mobile device downloads the latest output of $V^c, V_{u_i^p}$ and MLP_{user}, then the mobile device uses its local ratings data to calculate the gradients of $Server\_pt_{paras}$ and update $Device_{paras}$, then the server aggregates the gradients from all devices to update $Server\_pt_{paras}$ to complete a round of training. Servers and mobile devices continue to train until the loss stabilizes. When the pre-training phase is completed, the parameters of both the MF module, Meta module, MLP_{user} and MLP_{item} are fixed.

(2) Mapping stage. The Mapping stage is performed in the target domain. MLP_{user} firstly takes the train user's embedding in the source domain as input, and then the mapping network accepts the output of MLP_{user} as input, and outputs the predicted user's embedding in the target domain. The predicted embedding is used for rating prediction in the Prediction module. When the mapping phase is completed, the mapping network parameters are fixed. So far, all trainable parameters of MetaEM are trained.

(3) Cold-start stage. The cold-start stage generates accurate predicted ratings for cold-start users of the target domain. The scoring generation process is similar to the mapping stage. The cold-start user's embedding in the source domain is used as input, and the transformed embedding is generated through MLP_{user} and Mapping network. Then each mobile device downloads the predicted embedding and uses the local its Prediction module generates predicted ratings.

MetaEM can protect user privacy by saving user private data locally on the device. Typically, the strength of privacy protection depends on the content of the user gradient. MetaEM achieves a trade-off between privacy protection and recommendation performance. It deploys the module with the most model parameters on the server side, while the mobile device side only needs to deploy the rating prediction module with the least amount of parameters.

In the following sections, we introduce the implementation detail and technical process of each stage module of MetaEM.

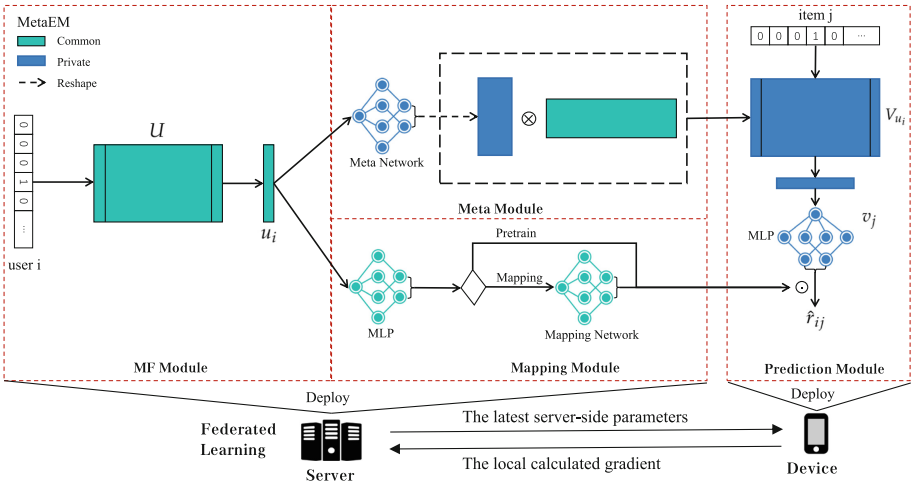


Fig. 1. An overview of MetaEM. It consists of four modules. The MF module generates user embeddings and common item embeddings, the Meta module generates private item embeddings for each user, the Mapping module transforms user embeddings from the source domain to the target domain, and the Prediction module is used to predict rating. The MF and Meta modules are deployed on the server (source and target) side, specifically, the Mapping module is deployed on the target server side, and the Prediction module is deployed on the mobile device side.

3.3 Pretrain Stage

Both the source and target domains need to obtain user and item embeddings information in the pretrain stage. The pretrain stage involves the MF, Meta and prediction modules, and the three modules cooperate to complete the entire pretrain process. The following subsections describe the design of each module in detail.

Matrix Factorization. The traditional matrix factorization algorithm decomposes the user rating matrix R into a user embedding matrix and an item embedding matrix, as shown in Eq. 1. R represents the rating matrix of $m \times n$, k is the dimension of the embedding, U represents the user embedding matrix of $k \times m$, where u_i represents the embedding of the user i. V represents the item embedding matrix of $k \times n$, where v_j represents the embedding of the item j. However, when the number of items is large, the scale of the item embedding matrix V will be very large. In the federated environment, the mobile devices need to continuously download the V and upload the gradient of the V, resulting in a serious communication load on the mobile device.

$$L = \min_{U,V} \left(\sum_i \sum_j \| I_{ij} \cdot \left(R_{ij} - u_i^T v_j \right) \|_F^2 \right) \tag{1}$$

To solve this problem, we propose to decompose the item embedding matrix into the product of the private item embedding matrix V^p and the common item embedding matrix V^c. The V^p is generated by a meta-network for each user. When V^c and V^p are transmitted to the mobile device, the amount of parameters is greatly reduced, thereby alleviating the communication load of the mobile device during the training process, as shown in Eq. 2, where $V^p_{u_i}$ denotes the private item embedding matrix of user i, V^c represents the common item embedding matrix shared by all users. Compared to directly generating V, which needs $O(k_s \times n)$ parameters, the MetaEM needs $O(k_s \times k_p + k_p \times n)$ parameters which $k_p < k_s$ and obviously reduces the item related parameters.

$$V_{u_i} = V^s \otimes V^p_{u_i} \tag{2}$$

Meta Recommemder Module. The meta-network takes the embedding of user i in source domain as input, and the outputs $V^p_{u_i}$. The definition of the meta-network is shown in Eq. 3, where $g(\cdot)$ is the meta-network, its parameter is ϕ, the input is the embedding of the user u_i, and the output dimension is $k_p \times k_s$ vector w_{u_i}, which is then reshaped into matrix $V^p_i \in \mathbb{R}^{k_c \times k_p}$.

$$w_{u_i} = g(u_i; \phi) \tag{3}$$

We define the meta-network as a multi-layer perceptron (MLP), where l represents the number of layers of the MLP, W_l and b_l are the weights and biases of the l layer, respectively.

$$\begin{aligned} &h_1 = ReLU(W_1 u_i + b_1), \\ &... \\ &h_l = ReLU(W_l h_{l-1} + b_l), \\ &h_{l+1} = ReLU(W_{l+1} h_l + b_{l+1}) \end{aligned} \tag{4}$$

Prediction Module. We generate the prediction model for each user, and define the prediction model as MLP_{user} and MLP_{item}, which are used to represent the nonlinear relationship between user embeddings and item embeddings, respectively. So we need to generate weights and biases for each layer. For the number of layers l, the weights and biases for this layer are $W^u_l \in \mathbb{R}^{f_{out} \times f_{in}}$ and $b^u_l \in \mathbb{R}^{f_{out}}$, where $f_i n$ represents the number of input layer parameters, f_{out} represents the number of output layer parameters. The outputs of MLP_{user} and MLP_{item} are calculated by Eqs. 5 and 6, respectively, and where l represents the number of layers, W_l and b_l are the weights and biases, u_i represents the embedding of user i, $\hat{u}_i$ represents the output of MLP_{user}, $V_{u_i,j}$ represents the embedding of item j, as shown in Eq. 2, and $\hat{v}_j$ represents the output of MLP_{item}.

$$\begin{aligned} &h_1 = ReLU(W_1 u_i + b_1), \\ &... \\ &h_l = ReLU(W_l h_{l-1} + b_l), \\ &\hat{U}_i = ReLU(W_{l+1} h_l + b_{l+1}) \end{aligned} \tag{5}$$

$$
\begin{aligned}
&h_1 = ReLU(W_1 V_{u_i,j} + b_1),\\
&\ldots\\
&h_l = ReLU(W_l h_{l-1} + b_l),\\
&\hat{v}_j = ReLU(W_{l+1} h_l + b_{l+1})
\end{aligned}
\tag{6}
$$

Then, we use the dot product of $\hat{U}_i$ and $\hat{V}_{u_i,j}$ as the predicted score for user u_i to item v_j, as shown in Eq. 7.

$$\hat{r}_{ij} = \hat{u}_i \odot \hat{v}_j \tag{7}$$

Loss. In order to learn the pretrain stage of MetaEM, we formulate the RP task as a regression problem and the loss function is defined as:

$$L_{rp} = \frac{1}{D_{train}} \sum_{r_{i,j} \in D_{train}} (r_{i,j} - \hat{r}_{i,j})^2 \tag{8}$$

To avoid overfitting, we add the $L2$ regularization term:

$$L_{reg} = \frac{1}{2} \|\Theta\|_2^2, \tag{9}$$

where Θ represents all the trainable parameters in the pretrain stage, including the user embeddings and item embeddings, the parameters of the meta network and the parameters of the prediction model.

The final loss L is a linear combination of L_{rp} and L_{reg}:

$$L = L_{rp} + \lambda L_{reg}, \tag{10}$$

where λ is the weight of L_{reg}. The whole procedure of pretrain stage can be efficiently trained using back propagation with federated learning on distributed mobile devices. The whole procedure of pretrain stage is shown in Algorithm 1.

3.4 Mapping Stage

The mapping stage is carried out in the target domain. After the pretrain of the source and target domains, the embedding sets of users and items can be obtained $\{ U^s, V^{cs}, g^s_\phi, U^t, V^{ct}, g^t_\phi \}$, contains the overlapped user and all item embeddings and meta-learners for the source and target domains. We use a MLP to capture the nonlinear relationship between source domain embeddings and target domain embeddings. In particular, we capture the relations between user embeddings transformed by MLP_{user}, as this can be conveniently used directly for recommendation tasks. The source domain server needs to share the embeddings of $\mathcal{U}^{train}$ to the target domain. The parameters of the mapping model are described as $Server\_mp_{paras} = \theta$. The mapping model is defined as follows:

$$\hat{u}^t_i = f_i(MLP_{user}(u^s_i); \theta) \tag{11}$$

Algorithm 1. Pretrain stage

Input: The user set $\mathcal{U}$, for user i, his local data D_i stored in his device; † means the code is executed in the server.

Output: $Server\_pt_{paras}$, $Device_{paras}$.

Initialize U, V^c, ϕ, MLP_{user} randomly in the server; Initialize MLP_{item} randomly in user i' device.
while not convergent **do**
  Sample a batch user S from $\mathcal{U}$.
  for user i in S **do**
    Calculate V^p based on g_ϕ;
    Download V^p, V^c from server; †
    Calculate the gradient of $Server\_pt_{paras}$ based on D_i; †
    Upload the gradient of $Server\_pt_{paras}$ to the server; †
    Update $Device_{paras}$ based on D_{u_i}; †
  end for
  Accumulate the gradient of $Server\_pt_{paras}$ gathered from S.
  Update $Server\_pt_{paras}$ based on the accumulated gradient.
end while
for user i in $\mathcal{U}^o$ **do**
  Calculate V^p based on g_ϕ;
  Deploy latest V^c, V^p in his device; †
end for

where the input to the mapping model is the output of MLP_{user}, $\hat{u}_i^t$ represents the transformed result of the embedding u_i^s.

To train the mapping model, we can optimize the distance between the input and output embeddings:

$$L = \sum_{user \in \mathcal{U}^{train}} \|\hat{u}_i^t - MLP_{user}(u_i^t)\|^2, \tag{12}$$

This optimization strategy is called distance-oriented, making $\hat{u}_i^t$ as close as possible to u_i^t.

However, since some users of the target domain have very few ratings, their embeddings may be unreasonable and inaccurate. Learning towards this inaccurate embeddings will adds negative impact to the mapping. Therefore, we adopt a task-oriented optimization strategy to directly use the final predicted rating as the optimization objective. The task-oriented loss function is defined as follows:

$$\hat{r}_{i,j} = \hat{u}_i^t \odot MLP_{item}(V_{u_i,j}) \tag{13}$$

$$L = \sum_{user \in D_{train}} (r_{i,j} - \hat{r}_{i,j})^2 + \lambda \frac{1}{2} \|\Theta\|_2^2 \tag{14}$$

where θ represents the parameters of the mapping model, λ represents the weight of the regularization term.

Compared with distance-oriented optimization, task-oriented can eliminate the influence of unreasonable user embedding, and have more training samples. The number of samples in the distance-oriented optimization is $|m|$, the number of samples in the task-oriented optimization is $|m \times n|$. The whole framework of cross-domain stage can be efficiently trained using back propagation with federated learning on distributed mobile devices. The whole procedure of mapping stage is shown in Algorithm 2.

Algorithm 2. Mapping stage

Input: The user set $\mathcal{U}^{train}$; For user i, his local data D_i^t and V^{ct}, V^p are stored in his device; † means the code is executed in the source App; ‡ means the code is executed in the target App.

Output: $Server\_mp_{paras}$

1: Initialize $Server\_mp_{paras}$ randomly in the target server.
2: **while** not convergent **do**
3: Sample a batch user S from $\mathcal{U}^o$.
4: **for** user i in S **do**
5: Calculate the gradient of $Server\_mp_{paras}$ based D_i^t and Prediction module; ‡
6: Upload the gradient of $Server\_mp_{paras}$ to the target server; ‡
7: **end for**
8: Accumulate the gradients of $Server\_mp_{paras}$ gathered from S.
9: Update $Server\_mp_{paras}$ based on the accumulated gradients.
10: **end while**

3.5 Cold-Start Stage

Since cold-start users do not have accurate embeddings in the target domain, we cannot recommend items of interest for them accurately. However, these users have rich ratings in the source domain, and the behavior of the same user in the source domain and the target domain is related. Based on this assumption, we can generate reasonable embedding for each cold-start user in the target domain, thus recommending items for them based on locally deployed Prediction module. Given the trained mapping model f_θ and embedding u_i^s, we can get the embedding of user in the target domain as shown in Eq. 11. Then, we can use the trained RP model to recommend items of interest to cold-start users, as shown in Eq. 14.

4 Experiments

We define three cross-domain tasks using the real-world Amazon review dataset and conduct sufficient experiments to answer the following questions:

- **Q1:** How does MetaEM perform in cold-start users comparing to the baseline cross-domain models?
- **Q2:** What is the contribution of meta-learner for the performance improvement?
- **Q3:** How does MetaEM perform in more practical scenarios of real-world recommendation?

4.1 Datasets

Following the existing methods [20,22], we adopted the large Amazon review dataset[1] for experiments. Specially, we use the Amazon-5scores dataset where the rating from 1 to 5.

Following [20,22], we choose three popular categories in total: Movie, Music and Book. We define three CDR tasks as Task 1: Movie → Music, Task 2: Book → Movie, and Task 3: Book → Music. The detailed statistics of the three datasets are shown in Table 2. In each CDR task, the number of ratings in the source domain is much larger than that in the target domain. There are many existing works that use only a small subset of the Amazon dataset, we use the entire dataset for experiments.

Table 2. Statistics about Amazon Dataset

CDR tasks	Domain		Item		User			Rating	
	Source	Target	Source	Target	Overlap	Source	Target	Source	Target
Task1	Movie	Music	5,0052	6,4443	1,8031	12,3960	7,6258	169,7533	109,7592
Task2	Book	Movie	36,7982	5,0052	3,7388	60,3668	12,3960	889,8041	169,7533
Task3	Book	Music	36,7982	6,4443	1,6738	60,3668	7,5258	889,8041	109,7592

4.2 Implementation Details

We use the Pytorch to implement our MetaEM and baselines. For each method, we use Adam optimizer and the initial learning rate are tuned within 0.0001, 0.001, 0.01, and set the regularization weight to 0.0001. Following [13,22], for all methods, We randomly select a fraction of overlapping users in the target domain and as test users, the other overlapping users are train users. We set the proportions of test users β as 20%, 50%, 80%. For all models, we set the batch size as 256. For all baselines, We set the dimension of user embeddings and item embeddings as 32. In addition, for each method, we report the mean results over ten random runs.

For MetaEM, we set the dimension of pravite item embeddings as 8 and common item embeddings as 32. The meta-network, mapping model and RP

[1] http://jmcauley.ucsd.edu/data/amazon.

model are defined as a three-layer network with hidden units 128. We use the *ReLU* function as the activation function. During training, we initialize all user embeddings and common item embeddings randomly with a normal distribution.

In our experiments, We implement CMF based on released code of the author[2]. We implement EMCDR based on released code of the author[3]. We implement PTUPCDR based on released code of the author[4]. If this paper is accepted, we will publish the codes of DCDCSR, SSCDR and MetaEM at github.

4.3 Baselines

Therefore, MetaEM belongs to the mapping-based method, so for fairness, we mainly choose the following methods as the baselines for comparison.

- TGT: TGT indicates that the target domain only uses its own data to complete MF training, which belongs to single-domain recommendation.
- CMF [16]: CMF achieves knowledge integration across multiple domains by connecting user rating matrices from multiple domains and sharing overlapped user embeddings across domains.
- DCDCSR [18]: This model continues the idea of EMCDR, but the learned mapping function is the mapping function from the target domain to the standard domain: the standard domain is a embedding space that fuses the source domain and the target domain together.
- SSCDR [6]: This model transforms the entire model into a semi-supervised learning process by designing an unsupervised loss function: it can utilize the rich non-overlapping entity data in the field, making the learned mapping function more robust and improving the recommendation performance.
- EMCDR [13]: EMCDR use MLP to capture the nonlinear mapping function between source domain and target domain, and propose only the users with sufficient data should be used to learn the mapping model, thus guaranteeing its effectiveness.
- PTUPCDR [22]. PTUPCDR uses a meta-learner to model a personalized preference bridge with user traits extracted from the user's interaction history in the source domain as input. Then use the personalized bridge to complete the mapping of the source domain embeddings to the target domain. This method is one the state-of-art method for cross-domain recommendation.

4.4 Evaluate Metrics

We adopt the recommendation scenario of explicit feedback of predicted ratings. We choose Root Mean Square Error (RMSE) and Evaluation Absolute Error

[2] https://github.com/VincentLiu3/CMF.
[3] https://github.com/masonmsh/EMCDR_PyTorch.
[4] https://github.com/easezyc/WSDM2022-PTUPCDR.

(MAE) as our evaluation metrics, as follows:

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|(y_i - \hat{y}_i)|$$
$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{15}$$

4.5 Experimental Results and Analysis

Cold-Start Experiments (Q1). This section presents the experimental results and analyzes the performance of MetaEM in cold-start scenario in details. We evaluate the effectiveness of MetaEM and the existing methods [6,13,16,18,22] on three cold-start scenarios under different values of β. The experiments are shown in Table 3 and the best result is marked with *. From the experimental results, we have the following findings:

(1) TGT is a single-domain recommendation model that only uses data from its own domain for training. Since cold-start users do not have enough ratings, their embedding quality is unsatisfactory. The recommended performance of TGT is unsatisfying. Compared with TGT, other cross-domain recommendation models use the rich data of the source domain to enrich the information of the target domain, so the cold-start users of the target domain can have better recommendation results. Therefore, the ability of single-domain recommendation to solve the cold-start problem is limited, and cross-domain recommendation performs better than single-domain recommendation in cold-start scenarios.
(2) CMF achieves knowledge integration across multiple domains by connecting user rating matrixcs from multiple domains and sharing overlapped user embeddings across domains. But it ignores that the embedding space between domains is different, and there is the problem of domain shift. So the recommendation result of CMF is better than that of single domain recommendation. On the contrary, the mapping-based models transform the embedding from source domain to the target domain, that can alleviates the influence of domain shift. Therefore, the way of cross-domain recommendation is very important for the recommendation results.
(3) EMCDR trains a mapping model that maps source-domain embeddings to target-domain, making full use of source-domain information to improve target-domain recommendation performance. So EMCDR outperform TGT and CMF. But EMCDR trains a common mapping model for all overlapping users, ignoring personalization among users. PTUPCDR trains a personalized mapping model for each overlapping user, and achieves better recommendation results than EMCDR. Therefore, we need to consider the differences between users when recommending.

(4) We find that MetaEM outperforms the best baseline model(PTUPCDR) in most scenarios, demonstrating the effectiveness of MetaEM on the cold-start problem. Unlike PTUPCDR, which trains a private mapping model for each user, MetaEM trains private item embeddings for each user. The private item matrix can represent the personalization of user interests. Consistent with PTUPCDR, MetaEM transfers the user embeddings from the source domain to the target domain. The difference is that in the cold-start phase, each cold-start user in the target domain uses the transferred embeddings to generate his private item matrix, getting better personalized recommendation results than PTUPCDR. In general, MetaEM belongs to personalization of interests, while PTUPCDR belongs to personalization of embedding mapping. Therefore, private item embeddings can perform better than private mapping models on the cold start problem.

Table 3. Experiment results (MAE & RMSE) for CDR tasks (the best result is marked with *)

	β	Metric	TGT	CMF	DCDCSR	SSCDR	EMCDR	PTUPCDR	MetaEM
Task1	20%	MAE	4.4765	1.5305	1.4869	1.3217	1.2364	1.1365	0.8138*
		RMSE	5.1374	2.0214	1.9052	1.6754	1.5574	1.1974	1.1226*
	50%	MAE	4.4886	1.6698	1.8023	1.3799	1.3201	1.2698	0.8394*
		RMSE	5.1985	2.2065	2.3341	1.7421	1.6568	1.6284	1.1359*
	80%	MAE	4.5095	2.4155	2.7021	1.5001	1.5009	1.4012	0.9373*
		RMSE	5.1845	3.0856	3.3069	1.9202	1.8765	1.8322	1.2051*
Task2	20%	MAE	4.1801	1.3622	1.4032	1.2399	1.1211	0.9972	0.8948*
		RMSE	4.7533	1.7898	1.7465	1.6549	1.4201	1.3319	1.1918*
	50%	MAE	4.2212	1.5765	1.6687	1.2136	1.1987	1.0874	0.9507*
		RMSE	4.8012	2.0598	2.0679	1.5618	1.5031	1.4232	1.2028*
	80%	MAE	4.2015	2.1487	2.3556	1.3088	1.3064	1.2012	1.1215*
		RMSE	4.8256	2.6508	2.7684	1.6979	1.6854	1.6031	1.4252*
Task3	20%	MAE	4.4812	1.8919	1.8319	1.5482	1.3602	1.2397	0.9897*
		RMSE	5.1564	2.3987	2.2916	1.9269	1.7015	1.6089	1.4116*
	50%	MAE	4.5179	2.1387	2.1862	1.4697	1.4697	1.3649	1.2299*
		RMSE	5.1849	2.7348	2.6597	1.8379	1.8006	1.7349	1.7335*
	80%	MAE	4.5316	3.0132	3.1467	1.6418	1.7095	1.5882	1.4468*
		RMSE	5.2465	3.7016	3.5947	2.1487	2.0948	2.0732	1.9648*

Explanation of the Performance Improvement (Q2). In this section, we conduct extensive experiments and analyze the contribution of private item embeddings, thereby explaining the improvement brought by MetaEM, and answer Q2.

We performed a visualization of target embeddings and transformed embeddings. We analyze the distribution of target embeddings and transformation embeddings in the embedding space. We further explore why MetaEM performs better than PTUPCDR from the embedding distribution, then demonstrate the capability of meta-network to generate private item embedding matrix for each user.

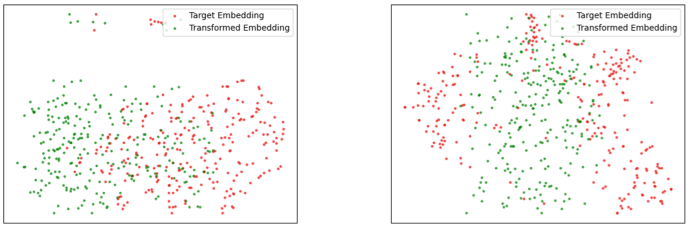
We use t-SNE [4] to visualize user embeddings learned by MetaEM and PTUPCDR on Task1 with $\beta = 0.2$. Figure 2(a) and 2(b) denote the embeddings of training and test users by PTUPCDR. Figure 2(c) and 2(d) denote the embeddings of trianing and test users by MetaEM. To be fair, we set the training and testing users for the MetaEM and PTUPCDR user visualizations to be the same. We selected 256 users from training users and test users for visualization, respectively. The red points represent the target embeddings (ground trouth) taken from target domain, while the green points denote the transformed embeddings.

Ideally, the t-SNE distribution of the transformed embeddings should be close to the target embeddings. From Fig. 2(a) and 2(b), We can see that there is a certain shift in the distributions of target embeddings and transformation embeddings. This shows that the recommendation results of PTUPCDR are not ideal. The main reason is that PTUPCDR multiplies the transformed embedding and item embedding to get the predicted score. This way is essentially a non-personalized matrix factorization. Therefore, PTUPCDR only realizes personalized transformation, but does not realize personalized recommendation.

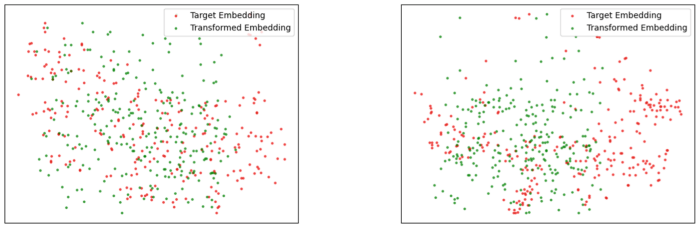
As shown in Fig. 2(c) and 2(d), we can find that the distribution of transformed embeddings by MetaEM can better fit the target embeddings distribution, demonstrating the capacity of the meta-network in MetaEM. The main reason is that the meta-network generates private item embeddings for each user, and the private item embeddings can represent the user's personalized preferences for items. After the cold-start user obtains the transformed embedding, the meta-network takes the transformed embedding as input, and generates the user's private item embeddings, so as to obtain the personalized preferences of cold-start users in the target domain. Therefore, we can recommend personalized items for cold-start users of the target domain, which could be the fundamental reason why MetaEM could achieve better overall performance.

Generalization Experiments Q(3). In this section, we perform extensive experiments to verify the compatibility of MetaEM on more complicated models.

Mapping-based cross-domain models are devoted to improving the quality of mapping models, and they mostly use MF to extract user and item embeddings. However, MF does not perform well for large-scale real-world recommendation. Therefore, we investigate the compatibility of MetaEM on two complicated models: GMF [2] and DNN [3]. For comparison, we follow [22], applying GMF and DNN on EMCDR and PTUPCDR. Unlike MF, which uses the vector inner product as the prediction, GMF uses the dot product of the user latent vector and the item latent vector as the output. DNN is a two-tower model. For GMF, the mapping function directly transforms the user embeddings of source domain. For DNN, the mapping function transforms the output of the user tower. We conduct generalization experiments on Task 1 with $\beta = 20\%$.



(a) PTUPCDR Train (b) PTUPCDR Test



(c) MetaEM Train (d) MetaEM Test

Fig. 2. The t-SNE visualization of MetaEM and PTUPCDR. We set the scenario as Task 1 and $\beta = 0.2$.

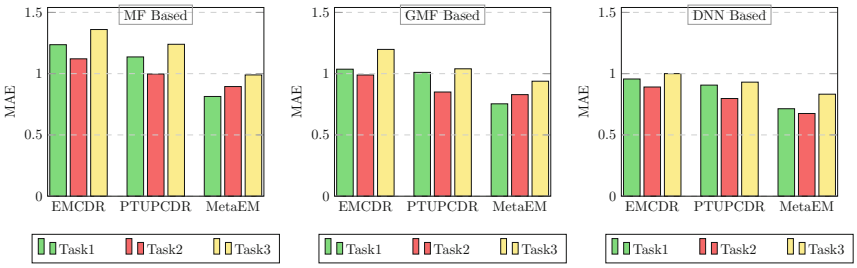


Fig. 3. Generalization experiments: using three base models MF, GMF, DNN for EMCDR, PTUPCDR and MetaEM, respectively. We set the scenario as Task 1 and $\beta = 0.2$

The generalization experiment results are shown in Fig. 3. We have several important observations: (1) The mapping-based cross-domain recommendation method is compatible with different embedding extraction models. EMCDR, PTUPCDR and MetaEM all significantly improve the recommendation performance on the target domain on GMF and DNN. Since DNN can extract more accurate embeddings than GMF, the improvement on DNN is higher than that of GMF, indicating that the quality of embedding can determine the recommendation performance of cross-domain models. (2) MetaEM has satisfying performance on different embedding models, which shows that MetaEM has good

compatibility with complicated models in large-scale real-word recommendation. Furthermore, the recommendation performance of MetaEM can constantly better than EMCDR and PTUPCDR when using the same embedding model.

5 Conclusion

The existing cross-domain recommendation models lack the protection of user privacy, and in the federated learning environment, federated cross-domain recommendation will bring communication load to mobile devices. To address the challenge of limited mobile device resources in federated cross-domain recommendation, we propose a light federated cross-domain recommendation model MetaEM based on embedding mapping. MetaEM is divided into pretrain stage, mapping stage and cold-start stage. Each stage follows the federated learning paradigm. In particular, we deploy a meta-learner on the server. It generates a private small-scale item embedding matrix for each user, thus reducing the communication load on mobile devices. To learn the mapping model stably, we employ a task-oriented optimization method. Experimental results on real-world datasets show that MetaEM has satisfying recommendation performance while well protecting user privacy data. In addition, MetmEM has well compatibility in different embedding models. MetaEM can play an important role in cross-domain recommendation c with limited mobile resources and a large number of items.

Acknowledgement. This work is supported by National Natural Science Foundation of China (62172155, 62072465).

References

1. Ammad-Ud-Din, M., et al.: Federated collaborative filtering for privacy-preserving personalized recommendation system. arXiv preprint arXiv:1901.09888 (2019)
2. Böhm, J., Niell, A., Tregoning, P., Schuh, H.: Global mapping function (GMF): a new empirical mapping function based on numerical weather model data. Geophys. Res. Lett. **33**(7) (2006)
3. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 191–198 (2016)
4. Donahue, J., et al.: DeCAF: a deep convolutional activation feature for generic visual recognition. In: International Conference on Machine Learning, pp. 647–655. PMLR (2014)
5. Hazrati, N., Shams, B., Haratizadeh, S.: Entity representation for pairwise collaborative ranking using restricted Boltzmann machine. Expert Syst. Appl. **116**, 161–171 (2019)
6. Kang, S., Hwang, J., Lee, D., Yu, H.: Semi-supervised learning for cross-domain recommendation to cold-start users. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1563–1572 (2019)
7. Kazama, M., Varga, I.: Cross domain recommendation using vector space transfer learning. In: RecSys Posters. Citeseer (2016)

8. Liang, F., Pan, W., Ming, Z.: FedRec++: lossless federated recommendation with explicit feedback. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4224–4231 (2021)
9. Lin, G., Liang, F., Pan, W., Ming, Z.: FedRec: federated recommendation with explicit feedback. IEEE Intell. Syst. **36**(5), 21–30 (2020)
10. Lin, Y., et al.: Meta matrix factorization for federated rating predictions. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 981–990 (2020)
11. Liu, J., Liu, X., Yang, Y., Wang, S., Zhou, S.: Hierarchical multiple kernel clustering. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI, pp. 2–9 (2021)
12. Liu, X., et al.: One pass late fusion multi-view clustering. In: International Conference on Machine Learning, pp. 6850–6859. PMLR (2021)
13. Man, T., Shen, H., Jin, X., Cheng, X.: Cross-domain recommendation: an embedding and mapping approach. In: IJCAI, vol. 17, pp. 2464–2470 (2017)
14. Mashhadi, M.B., Shlezinger, N., Eldar, Y.C., Gunduz, D.: FedRec: federated learning of universal receivers over fading channels (2020)
15. Muhammad, K., et al.: FedFast: going beyond average for faster training of federated recommender systems. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1234–1242 (2020)
16. Singh, A.P., Gordon, G.J.: Relational learning via collective matrix factorization. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 650–658 (2008)
17. Zhang, Y., Cao, B., Yeung, D.Y.: Multi-domain collaborative filtering. arXiv preprint arXiv:1203.3535 (2012)
18. Zhu, F., Wang, Y., Chen, C., Liu, G., Orgun, M., Wu, J.: A deep framework for cross-domain and cross-system recommendations. arXiv preprint arXiv:2009.06215 (2020)
19. Zhu, F., Wang, Y., Chen, C., Liu, G., Zheng, X.: A graphical and attentional framework for dual-target cross-domain recommendation. In: IJCAI, pp. 3001–3008 (2020)
20. Zhu, Y., et al.: Transfer-meta framework for cross-domain recommendation to cold-start users. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1813–1817 (2021)
21. Zhu, Y., et al.: Transfer-meta framework for cross-domain recommendation to cold-start users, pp. 1813–1817. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3404835.3463010
22. Zhu, Y., et al.: Personalized transfer of user preferences for cross-domain recommendation. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, pp. 1507–1515 (2022)