Meta-learned ID Embeddings for Online Inductive Recommendation

Jingyu Peng, Le Wu(✉), Peijie Sun, and Meng Wang

School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China

Abstract. Learning accurate user and item ID embeddings from user-item historical records has shown great success for recommender systems. Most of these embedding learning models are transductive and work well for users that appear in the training stage. However, in the model serving stage, new users continue to join the system. It's important to quickly adapt to new users' preferences for online inductive recommendation scenarios. Some previous works adopted embedding retraining or fed content data for new user ID embedding learning, these models either suffered from slow convergence or relied on auxiliary data. In this paper, we propose a meta-learned ID embedding framework for new users without using any side information in online inductive recommendation scenarios. Our key idea is that, we treat each user's ID embedding learning as a separate task, and propose to meta-learn the initial embedding by modeling the global knowledge from all users (tasks). Each user's embedding is initialized by the learned global knowledge instead of randomly initialization. Therefore, we could quickly adapt to a new user's ID embedding based on a few updates from her online records, which can facilitate fast online recommendation. Moreover, our main technical contribution lies in how to learn the global prior knowledge for informative ID embedding initialization without any side information. Finally, extensive experimental results on three real-world datasets clearly show both the efficiency and effectiveness of the meta-learned ID embeddings for inductive recommendation.

Keywords: Recommender system · Meta learning · Collaborative Filtering · Cold-start

1 Introduction

Collaborative Filtering (CF) is a popular approach for building recommender systems, with the assumption that users' preferences for items could be collaboratively modeled from users' historical behavior data [1,20]. Among all CF models, learning accurate user and item ID embeddings has been the key technology that dominates CF area [3,15,18,24,25]. These embedding models can learn low dimensional dense vector representations of users from their past behavior. However, most of these ID embedding based models are naturally transductive, meaning that during the model serving process, each test user must appear in the training process. In practical recommender systems, new users continuously join the platform, e.g., a new user registers to or an anonymous user enters a platform, and shows preferences to some items (e.g., browses or buys several items). It is very cruical to update recommender systems timely to serve these new users, as it can improve user satisfaction and increase their loyalty to the platform.

To achieve the inductive learning with new users at test time, some researchers provided content-based approaches to learn new users' preferences, but most new or anonymous users are reluctant to fill any personal information [6,22]. Another naive idea is to retrain recommender models with new users' behavior. As full retraining is time consuming, an alternative solution is only to update new users' embeddings while keeping the embeddings of items and old users fixed. For each new user, the training process usually starts with random initial user embedding, and the model performance needs many update times to reach a local minimum. Although this fine-tune process shows comparable efficiency compared to the full data retraining, obtaining new users' final ID embedding with many training epochs is still far from the online latency requirements. In summary, relying on auxiliary data or suffering from the time efficiency issue make current models inferior choices for serving new users online.

In this paper, we explore whether it is possible to provide fast recommendation for new users in inductive recommendation without using any side information. Instead of randomly initializing ID embeddings for new users, our high-level idea is to learn better initial embeddings for new users, such that to speed up the learning process of new users with very limited records. As predicting each user's preference to an item can be regarded as a classification problem, we treat each user's embedding learning as a separate task, and make an analogy between recommending some products to a new user with few interactions and few-shot classification [5,12]. Therefore, it is natural to apply meta-optimization approaches, which are successful in fast adaption of few-shot classification [5]. The core idea of meta-optimization approaches is to train global sharing initialization parameters for all tasks (users). When a new user (task) comes, her ID embedding could be initialized with the global learned knowledge, then her final ID embedding can be quickly to be adapted with a few updates to facilitate fast online recommendation.

With the analogy between recommending some products to a new user with few interactions and few-shot classification, there are several recent attempts that leveraging meta-optimization approaches to generate better initial embeddings for new users, either with the entity profile information [11,17] or with auxiliary heterogeneous information networks [13]. Nevertheless, it is non-trivial to apply these meta-optimization techniques, as we do not have any content input for new users. How to define the general sharing initialization parameters for all users becomes the key challenge. To tackle this challenge, we design two detailed strategies for global parameters initialization. The first model is straightforward by treating the initialization ID embedding as global parameter for all users, i.e., all users share the same initial ID embedding. The second idea is feeding pretrained item embeddings and the current user's limited records to learn the global parameters that can be used to output the unique ID embedding of each user. Finally, we conduct extensive experiments on three real-world datasets and the experimental results clearly show the effectiveness and efficiency of our proposed framework. For example, our proposed framework could improve the recommendation accuracy with more than 10% and the training efficiency with less than one-tenth time cost compared to the best baseline.

2 Preliminaries

Given the user set $\mathcal{U} = (1, 2, 3, ..., M), |\mathcal{U}| = M$ and item set $\mathcal{V} = (1, 2, 3, ..., N), |\mathcal{V}| = N$, their interaction records form the rating matrix $\mathbb{R}^{M \times N}$. In this matrix, r_{uv} equals one when user u rates item v but zero otherwise. For each user u, we use $\mathcal{R}_u^+$ to denote the positive itemset that u shows preferences, i.e., $\forall v \in V, v \in \mathcal{R}_u \iff r_{uv} = 1$. And we randomly select k times the size of $\mathcal{R}_u^+$ items from the set $\mathcal{V} - \mathcal{R}_u^+$. The selected items are treated as the negative itemset $\mathcal{R}_u^-$.

With users' preferences, state-of-the-art CF embedding models focus on learning a low dimensional embedding space of users and items, namely $\mathbf{P} \in \mathbb{R}^{D \times M}$ and $\mathbf{Q} \in \mathbb{R}^{D \times N}$. Then, the predicted preference of the user-item pair (u, v) is modeled by the inner product of their corresponding embedding vectors as:

$$\hat{r}_{uv} = \mathbf{p}_u^T \mathbf{q}_v. \tag{1}$$

The widely used binary cross-entropy loss [7] is adopted as the loss function as:

$$\mathcal{L} = -\sum_{u=1}^{M} \sum_{i \in \mathcal{R}_u^+} \sum_{j \in \mathcal{R}_u^-} (r_{ui} \log(\hat{r}_{ui}) + (1 - r_{uj}) \log(1 - \hat{r}_{uj})) + \lambda ||\mathbf{P}||_F^2 + \lambda ||\mathbf{Q}||_F^2, \tag{2}$$

where the first term captures the training loss, the last two terms are l2-norm regularization terms with model parameters as $[\mathbf{P}, \mathbf{Q}]$, and λ is a regularization parameter.

These CF models perform well for the transductive setting, i.e., all test users appear in the training data. However, in the real world, inductive learning is more general with new users continuously join the system and show their preferences with very limited records in a short session. How to provide timely recommendations for new users has become a critical issue. A straightforward solution is to retrain the embedding-based recommendation model with both the online new user data and offline data. However, the retraining time is time-consuming. An alternative solution is to only learn the ID embeddings of new users and keep learned item embeddings fixed as the item embeddings have been well trained offline. Let a denote a new user that does not appear offline, i.e., $a \notin \mathcal{U}$, our goal is to learn the new user ID embedding $\mathbf{p}_a$ by minimizing the following function:

$$\mathcal{L}_a = -\sum_{i \in \mathcal{R}_a^+} \sum_{j \in \mathcal{R}_a^-} (r_{ai} \log(\hat{r}_{ai}) + (1 - r_{aj}) \log(1 - \hat{r}_{aj})) + \lambda ||\mathbf{p}_a||_F^2, \tag{3}$$

where $\mathbf{p}_a$ is the embedding of new user a that does not appear in the training data, and λ is the same regularization parameter as Eq. (2).

Please note that, as each new user has very limited records, the average number of most new users' rating records is far less than the ID embedding size D, i.e., $|\mathcal{R}_a^+| \ll D$. Given the optimization function for each new user a, we start with a random initialization of $\mathbf{p}_a$, and perform gradient descent until convergence. The training process will cost hundred of update epochs. Thus, it could not satisfy users' real-time needs for serving new users online. It is natural to ask the question: could we design a fast learning model for new users, such that we could quickly learn a new user's preference with a few gradient steps?

3 Meta-learned ID Embeddings for New Users

In this section, we first propose how to recast the problem of fast inductive recommendation with new users under the meta-learning framework. Then, we give two detailed architectures of designing the meta-learned ID embedding models without any content input. After that, we briefly show how to quickly adapt to new users' ID embeddings at online serving stage with the learned meta-knowledge.

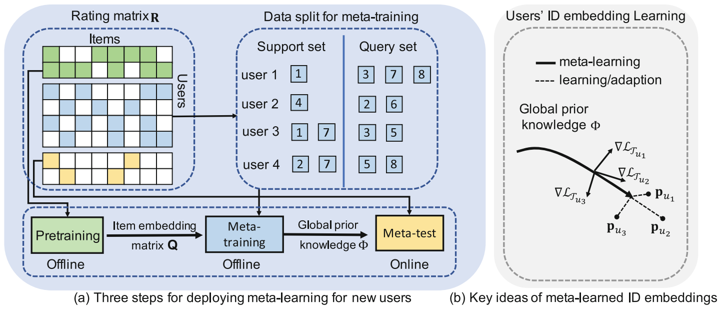

Fig. 1. The overall framework of our proposed framework, with the key idea of meta-learned ID embedding is shown at the right part of this figure.

3.1 Meta-learning for Inductive Recommendation

To quickly learn each new user's embedding vector with her limited rating records, we could build a connection between meta-learning and inductive recommendation for new users. By treating each user u's ID embedding learning as a task, each task (user) has very limited training data $\mathcal{R}_u^+$. Meta-learning provides a potential solution to our problem: by learning to learn across data from many previous tasks (users), meta-learning algorithms can discover the global meta-knowledge among tasks (all training users) to enable fast learning on new tasks (new users). In our framework, the prior knowledge is denoted as a parameter set Φ, and instead of random initialization of each user's embedding $\mathbf{p}_u^0$ as previous works, we learn a function g parameterized by Φ to initialize ID embedding vector $\mathbf{p}_u^0 = g(;\Phi)$.

For ease of clarification, in Fig. 1, we show the concrete steps for deploying meta-learning for new users. There are three steps: pretraining, meta-training and meta-test. These three steps follow a natural time line. In the pretraining step, we can adopt any embedding models to learn user and item embeddings. This step outputs item embedding matrix $\mathbf{Q}$ for the following two steps. Next, we mimic the meta-training process to divide the data of each task into a support set and a query set, and design meta-learning framework to learn the global knowledge. The global knowledge is then sent to the online stage to initialize the new user's ID embedding, which can facilitate quickly adapting to new users' ID embedding learning through one or more gradient steps.

Specifically, during meta training, similar to the setting in MAML, we split the original training data of each task $\mathcal{T}_u$ into two sets: a support set $\mathcal{S}_u$ and a query set $\mathcal{Q}_u$. The support set and query set come from u's rated itemset $\mathcal{R}_u^+$ and are mutually exclusive: $\mathcal{S}_u \cap \mathcal{Q}_u = \emptyset, \mathcal{S}_u, \mathcal{Q}_u \subseteq \mathcal{R}_u^+$. Each task $\mathcal{T}_u$ is associated with task-specific local parameters, i.e., the ID embeddings $\mathbf{p}_u$. We use $\mathbf{p}_u^0$ to denote the initial embedding of the local parameter $\mathbf{p}_u$. Then, this task updates its local parameter $\mathbf{p}_u$ with the support set $\mathcal{S}_u$ using one or a few gradient steps. For example, when using one gradient update, we have:

$$\begin{aligned}\mathbf{p}'_u &= \mathbf{p}_u - \alpha \nabla_{\mathbf{p}_u} \mathcal{L}_{\mathcal{T}_u}(g(;\Phi)) \\ &= \mathbf{p}_u - \alpha \nabla_{\mathbf{p}_u} \mathcal{L}_{\mathcal{T}_u}(\mathbf{p}_u^0),\end{aligned} \tag{4}$$

where α is the step size parameter for local parameter update, and $\mathcal{L}_{\mathcal{T}_u}$ is the loss function with regard to task $\mathcal{T}_u$. Without loss of generality, we use the cross-entropy loss as:

$$\mathcal{L}_{\mathcal{T}_u} = -\sum_{i \in \mathcal{S}_u} \sum_{j \in \mathcal{S}_u^-} (r_{ui} log(\hat{r}_{ui}) + (1 - r_{uj}) log(1 - \hat{r}_{uj})), \tag{5}$$

where $\mathcal{S}_u^-$ is k times of the size of $\mathcal{S}_u$, and $\mathcal{S}_u^- \subseteq \mathcal{R}_u^-$. In the above equation, we do not have any regularization term as Eq. (3). The reason is that, meta-learning algorithms only perform several gradient steps based on Eq. (5), and works as a early stopping without any overfitting issue.

For each task $\mathcal{T}_u$, after learning the updated local parameter $\mathbf{p}'_u$, we learn how to learn the performance of global parameters Φ with the query set $\mathcal{Q}_u$. The global parameters Φ is trained by optimizing the performance of updated local parameters $\mathbf{p}'_u$ with respect to Φ across all tasks (users).

$$\arg\min_{\Phi} \sum_{u \in \mathcal{Q}_u} \mathcal{L}_{\mathcal{T}_u}(\mathbf{p}_u^{'}) = \mathcal{L}_{\mathcal{T}_u}(\mathbf{p}_u - \alpha \nabla_{\mathbf{p}_u} \mathcal{L}_{\mathcal{T}_u}(g(;\Phi))). \tag{6}$$

In the above equation, please note that the meta-optimization performance is evaluated on the updated task-specific local parameters $\mathbf{p}_u^{'}$, which are learned from current global parameters Φ (e.g., a step update with Eq. (4)). Then, meta-optimization over tasks (users) is also updated with stochastic gradient descent as:

$$\Phi \leftarrow \Phi - \beta \nabla_{\Phi} \sum_{u \in \mathcal{Q}_u} \mathcal{L}_{\mathcal{T}_u}(\mathbf{p}_u^{'}), \tag{7}$$

where β is the meta step size.

3.2 Architecture of Meta-Learned ID Embeddings

Given the formulation above, the problem of meta-learned ID embedding framework turns to how to build a function $g(;\Phi)$ to extract global knowledge structure for initial user ID embedding of each task $\mathcal{T}_u$ as: $\mathbf{p}_u^0 = g(;\Phi)$.

General Learner. Without any side information as input, a simple idea of the global knowledge learner is to set g to an identity function. Formally, for any task $\mathcal{T}_u$ of a user u, we have:

$$\mathbf{p}_u^0 = g_1(;\Phi) = \mathbf{I}\Phi = \Phi, \tag{8}$$

where $\mathbf{I} \in \mathbb{R}^{D \times D}$ is an identity matrix, and $\Phi \in \mathbb{R}^D$. In other words, we assume that there exists similarities of all users, and it is presented in the form that each user has a same preference initialization vector.

Personalized Learner. The general learner is simple, but its expressiveness may be limited by assigning the same initialization vector for all users. As each user u has limited available records $\mathcal{S}_u$, we design a personalized embedding learner to fully utilizing her rating records. Since we already pretrained item embedding matrix $\mathbf{Q}$, the personalized initialization vector for each user can be calculated with:

$$\mathbf{x}_u = Pooling(\mathbf{Q}[\mathcal{S}_u]) \tag{9}$$

$$\mathbf{p}_u^0 = g_2(\mathbf{x}_u;\Phi), \tag{10}$$

where $\mathbf{Q}[\mathcal{S}_u]$ denotes the sub item embedding matrix As each user's support set varies, the pooling operation in Eq. (9) transforms the variable length submatrix into a fixed size vector output. Both the pooling function and the learner g_2 can be flexible. In practice, we choose average pooling as it achieves better performance compared to max pooling. Equation (10) could be a linear function as $\mathbf{p}_u^0 = \mathbf{W}\mathbf{x}_u$ with transformation matrix $\mathbf{W}$, or a multilayer perceptron to capture the non-linear relationships. Compared to the general learner, the personalized learner utilizes more personalized information for initial embedding learning.

Algorithm 1. Training Process of Meta-Learned ID Embedding

```
Input: Task $\mathcal{T}_u$ with support and query set
Input: Pretrained item-embedding matrix $\mathbf{Q}$
Input: Step size hyperparameters $\alpha$ and $\beta$
Input: The local update times $K$
Output: The shared global parameters $\Phi$
 1: Randomly initialize global parameter $\Phi$
 2: while Not converge do
 3:     Randomly sample batch of users $\mathcal{B} \subset \mathcal{U}$
 4:     for user $u$ in $\mathcal{B}$: do
 5:         Initialize $\mathbf{p}_u^0 = g(;\Phi)$ based on a detailed architecture;
 6:         $\mathcal{L} = 0$;
 7:         for $k = 1; k \leq K; k++$ do
 8:             for $v \in V$: do
 9:                 $\hat{r}_{uv} = \mathbf{q}_V^T \mathbf{p}_u^{k-1}$;
10:             end for
11:             Calculate loss $\mathcal{L}_u$ based on Eq.(5);
12:             $\mathbf{p}_u^k \leftarrow \mathbf{p}_u^{k-1} - \alpha \nabla_{\mathbf{p}_u^{k-1}} \mathcal{L}_{\mathcal{T}_u}$;
13:         end for
14:         $\mathbf{p}_u = \mathbf{p}_u^K$;
15:         for $v \in V$: do
16:             $\hat{r}_{uv} = \mathbf{q}_V^T \mathbf{p}_u$;
17:         end for
18:         Calculate $\mathcal{L}_u$ with Eq.(5) based on query set $\mathcal{Q}_u$;
19:         $\mathcal{L} = \mathcal{L} + \mathcal{L}_{\mathcal{T}_u}$
20:     end for
21:     $\Phi \leftarrow \Phi - \beta \nabla_\Phi \mathcal{L}$;
22: end while
23: Return Global parameter set $\Phi$.
```

We show the details of meta-training in Algorithm 1. For the above two detailed architectures, the only difference in this algorithm is calculating $g(;\Phi)$ in Line 5. In practice, for each user u, the unobserved feedbacks $\mathcal{V} - \mathcal{S}_u$ is much larger than the observed support set $\mathcal{S}_u$ in Eq. (5). Similar as many previous works [3,7,19], we randomly select 3 times of the size of $\mathcal{S}_u$ as possible negative items at each training epoch.

3.3 Meta-test Stage

After finishing the meta-training process, we get the global parameter set Φ. For online serving stage, if a new user a comes and shows preferences to a limited item set $\mathcal{S}_a$, we could initialize her embedding as $\mathbf{p}_a^0 = g(;\Phi)$ and quickly update her embedding with K gradient steps.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct experiments on three real-world datasets: *MovieLens-1M*[1], *Amazon Cell Phones and Accessories* and *Amazon CDs and Vinyl*[2]. In the following subsections, the MovieLens-1M, Amazon Cell Phones and Accessories, and Amazon CDs and Vinyl are called MovieLens, Amazon Small and

[1] https://grouplens.org/datasets/movielens/1m/.
[2] http://jmcauley.ucsd.edu/data/amazon/.

Table 1. The statistics of the three datasets.

Datasets		MovieLens	Amazon small	Amazon Big
Pre-training	Users	1,510	6,970	18,815
	Items	3,952	9,448	57,750
	Ratings	249,088	59,606	252,706
Meta-training	User	3,020	13,937	37,626
	Ratings	18,316	59,606	182,832
	Avg size of support set	2.81	1.93	2.21
	Avg size of query set	3.25	2.34	2.65
Meta-test	Users	1,510	6,953	18,749
	Ratings	8,978	33,497	97,640
	Avg size of support set	2.70	2.15	2.35
	Avg size of query Set	3.20	2.67	2.85

Amazon Big for short. As we focus on the ranking task, for all datasets, we transform the ratings in the original datasets into implicit feedback. If one user rates an item, the corresponding entry will be treated as 1, otherwise it will be 0. As illustrated in Fig. 1, there are three steps in deploying meta-learning for new users: pretraining, meta-training, and meta-test. Since there should be no duplicate users among these three steps, we randomly split all users into three parts in the ratio 1:2:1 for pretraining, meta-training, and meta-test, respectively. In the pretraining stage, we randomly select one record of each user as the validation data. In the meta-training stage, for each user we randomly select 2 to 10 historical records of her. Half of the selected records are treated as the support set, and the rest of the selected records are treated as the query set. And we adopt the same procedure to prepare the support set and query set for each user in the meta-test stage. Details of all dataset are shown in Table 1.

Experimental Setup. We call our proposed framework of meta-learned ID embeddings MetaCF for inductive CF. The two detailed architectures for MetaCF are denoted as MetaCF_G for general embedding (Eq. (8)) and MetaCF_P for personalized embedding (Eq. (10)). As (10) could be a linear function or a multilayer perceptron to capture the non-linear relationships, we use MetaCF_P(Linear) and MetaCF_P(Neural) to denote these two choices. The neural architecture is a two-layered neural network with ReLU activation. To study the efficiency and effectiveness of our proposed model, the classical recommender model Bayesian Personalized Ranking [19] and the meta-learning based model MeLU [11] are chosen as the baseline models. For fair comparison, the two baselines use the same pretrained item embedding matrix as our proposed framework. They start with random initialization of new users with meta-test data, and use the same prediction function as our proposed framework with the similar loss function (Eq. (3)). The performance of all models is evaluated on the query set of the users in the meta-test stage. The meta-learning based models, i.e., our proposed model and MeLU, are trained on the meta-train data first.

Then for each new user in the meta-test stage, the support set of each user is used to learn her ID embedding. However, BPR is only trained on the support set of all users in the meta-test stage to learn the ID embeddings of all users. Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [26,27] are used to evaluate the performance of all models. For both metrics, a larger value means a better performance.

Parameter Setting. For all models, we set the dimension D to 32. The step size alpha is set to 2×10^{-3} for all models and beta is set to 1×10^{-7}, 1×10^{-6}, and 1×10^{-5} for MetaCF_G, MetaCF_P(Linear), and MetaCF_P(Neural), respectively. The local updates times K varies from one to five. Please note that, as the meta-gradient involves second derivatives when performing back gradient over the meta-objective (Eq. (5)), we resort to first-order approximation, which shows nearly the same performance as obtained with full second derivatives [5].

Table 2. Overall performance of our proposed models. Bold font means the best model and underline means the corresponding model ranks second.

Models	MovieLens		Amazon small		Amazon Big	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
BPR_5	0.0341	0.0258	0.0195	0.0130	0.0047	0.0031
BPR_Best	<u>0.0656</u>	0.0462	<u>0.0308</u>	**0.0209**	0.0070	0.0045
MeLU	0.0220	0.0108	0.0048	0.0029	0.0022	0.0013
MetaCF_G	0.0444	0.0350	0.0247	0.0178	0.0066	0.0042
MetaCF_P(Linear)	0.0565	<u>0.0465</u>	**0.0312**	0.0203	**0.0092**	**0.0060**
MetaCF_P(Neural)	**0.0694**	**0.0478**	0.0298	<u>0.0205</u>	<u>0.0081</u>	<u>0.0053</u>

4.2 Model Performance

We report the overall performance comparison of our proposed framework and baselines in Table 2. To verify whether our proposed models can quickly adapt to a new user's ID embedding based on a few updates from her online records, we report the results of our proposed models when the local update times is set as 5. The local update times of MeLU is also set to 5. We report two kinds of results of the BPR model. BPR_5 denotes when the training epoch is set as 5. And BPR_Best denotes the best performance of BPR without training epoch limitation.

According to the results in Table 2, we have the following conclusions. First, compared with BPR_5 and MeLU, all our proposed models have significant improvements on three datasets under all the metrics. The epochs of the BPR_Best for MovieLens, Amazon Small, and Amazon Big are 44, 49, and 50, respectively. Although the local update times of our proposed model is set to 5, our proposed models perform better than BPR_Best on MovieLens and Amazon

Big datasets. E.g., on Amazon Big dataset, the NDCG@10 reaches 0.0060 for MetaCF_P(Linear), with more than 10% improvement compared to BPR_Best. We guess a possible reason of the recommendation performance gain is that, as MetaCF_P(Linear) can learn the global knowledge to speed up training for new users and the prior knowledge of all training users can help to alleviate the extreme sparsity of test users. The reason why MeLU performs worse may be that MeLU is designed for the content-based recommendation without considering any collaborative information.

When comparing the performance of our proposed three architectures, we observe that MetaCF_G with the same meta-learned initialization of all users could already reach quite good results. MetaCF_P(Linear) and MetaCF_P(Neural) perform better than MetaCF_G on MovieLens and Amazon Small datasets. By comparing the performance of MetaCF_P(Linear) and MetaCF_P(Neural), we can find the neural implementations achieve better result on Movielens and Amazon Small, while the linear implementations perform better on Amazon Big. In despite of the stronger express ability of neural architecture, the data sparsity and data size limit the performance.

Table 3. Performance with different local update times K under metrics HR@10.

Model		Local update times K				
		1	2	3	4	5
MovieLens	BPR_K	0.0110	0.0168	0.0236	0.0298	0.0341
	MeLU	0.0213	0.0224	0.0235	0.0219	0.0220
	MetaCF_G	0.0264	0.0341	0.0395	0.0429	0.0444
	MetaCF_P(Linear)	0.0543	0.0546	0.0560	0.0559	0.0565
	MetaCF_P(Neural)	0.0677	0.0682	0.0686	0.0691	0.0694

In Table 3, we show the performance of all models with different values of local update times K on Movielens. As K ranges from 1 to 5, we find the performance of BPR_K has a large improvement. By contrast, the performance of our proposed models is relatively stable and is not strongly influenced by K. We think this is caused by the meta-knowledge learned by our proposed models, such that one local update can already reach very good ID embedding. Based on this result, in practice with very high time request, we could set the local update times to 1.

In Fig. 2, we compare the performance of all models with different support set size on Amazon_Big. From the result, we can find the performance of all models improves stable with increasing the support set size. And under all cases, our proposed models always perform better than the baselines.

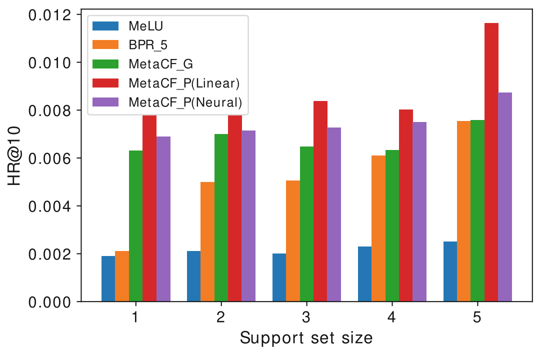

Fig. 2. Performance with different support set size under metrics HR@10 on Amazon_Big

5 Related Work

Learning low-dimensional ID embeddings of users and items from their historical behavior has been proved extremely useful for modern recommender system design [10,19]. Most CF approaches are transductive and could not apply to new nodes at test stage. To tackle the new node problem at test stage, some works are proposed to leverage node content to build a connection between content embedding and ID embedding for inductive learning [6,22]. Researchers have recently attempted to utilize sub-graph based neural network between each possible user-item pair for inductive matrix factorization with new users or items [28]. However, due to the huge time complexity of each candidate sub-graph modeling, it is impractical for online stage. To tackle the streaming data problem at model serving stage, how to incrementally update and retraining these systems is also a hot topic [8,29]. For new users at test stage, a simple idea is to keep the item embeddings learned from history fixed, while learning new user ID embedding from random initialization [9,23]. In practice, these models still cost many epoches to reach stable. Different from these works, we focus on how to quickly adapt to each new user's ID embedding with a better ID embedding initialization.

Meta-learning, based on "learning to learn" concept, learns the meta-knowledge through a variety of learning tasks [5,12,16]. Among all meta-learning approaches, Model-agnostic Meta-Learning (MAML) is an optimization based meta-learning approach that is widely used in many scenarios [5]. MAML treats the learned shared global parameter as the initial state of any task, such that the local parameters of each new task can be achieved with very few gradient steps and a small of amount of data. Meta-learning models are employed in various recommendation scenarios. Most meta-learning based approaches for recommendation focused on the cold-start recommendation with user or item entity features [2,11,17,21,30], auxiliary heterogeneous networks [13], or the sparse context data [4]. E.g., Pan et al. proposed an optimization-based method to learn an ID generator to generate desirable initial embeddings for new ad IDs based on the features of ads [17]. And MeLU is designed for content based recommendation with user preference estimator is trained with meta-learning[11]. Besides, meta-learning approaches are also used to select user-level adaptive recommendation model selection [14]. We differ greatly as we focus on fast adaption of user embeddings without any content or auxiliary data, which makes our model more general in practice.

6 Conclusions and Future Work

In this paper, we proposed a meta-learning framework for online inductive setting with new users. To the best of our knowledge, we are one of the first few attempts that provided meta-learned new user ID embedding without any content information. By recasting this problem as a meta-learning solution, we designed different architectures to transform the prior knowledge into initial ID embeddigns without any content input. Extensive experimental results on three real-world datasets clearly showed the effectiveness and efficiency of our proposed framework. In the future, we would like to design meta-learning algorithms for online inductive recommendation with both new users and new items.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. TKDE **17**(6), 734–749 (2005)
2. Bharadhwaj, H.: Meta-learning for user cold-start recommendation. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2019)
3. Chen, L., Wu, L., Hong, R., Zhang, K., Wang, M.: Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach. In: AAAI, vol. 34, pp. 27–34 (2020)
4. Du, Z., Wang, X., Yang, H., Zhou, J., Tang, J.: Sequential scenario-specific meta learner for online recommendation. In: SIGKDD, pp. 2895–2904 (2019)
5. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML, pp. 1126–1135 (2017)
6. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS, pp. 1024–1034 (2017)
7. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: WWW, pp. 173–182 (2017)
8. Huang, X., Wu, L., Chen, E., Zhu, H., Liu, Q., Wang, Y.: Incremental matrix factorization: a linear feature transformation perspective. In: IJCAI, pp. 1901–1908 (2017)
9. Jiang, M., Cui, P., Wang, F., Zhu, W., Yang, S.: Scalable recommendation with social contextual information. IKDE **26**(11), 2789–2802 (2014)
10. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: SIGKDD, pp. 426–434 (2008)
11. Lee, H., Im, J., Jang, S., Cho, H., Chung, S.: MeLU: meta-learned user preference estimator for cold-start recommendation. In: SIGKDD, pp. 1073–1082 (2019)

12. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-SGD: learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835 (2017)
13. Lu, Y., Fang, Y., Shi, C.: Meta-learning on heterogeneous information networks for cold-start recommendation. In: SIGKDD, pp. 1563–1573 (2020)
14. Luo, M., et al.: MetaSelector: meta-learning for recommendation with user-level adaptive model selection. In: WWW, pp. 2507–2513 (2020)
15. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. In: NIPS, pp. 1257–1264 (2008)
16. Nichol, A., Schulman, J.: Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999 **2**(3), 4 (2018)
17. Pan, F., Li, S., Ao, X., Tang, P., He, Q.: Warm up cold-start advertisements: improving CTR predictions via learning to learn id embeddings. In: SIGIR, pp. 695–704 (2019)
18. Rendle, S.: Factorization machines. In: ICDM, pp. 995–1000 (2010)
19. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461 (2009)
20. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW, pp. 285–295 (2001)
21. Vartak, M., Thiagarajan, A., Miranda, C., Bratman, J., Larochelle, H.: A meta-learning perspective on cold-start recommendations for items. In: NIPS, pp. 6904–6914 (2017)
22. Volkovs, M., Yu, G., Poutanen, T.: DropoutNet: addressing cold start in recommender systems. In: NIPS, pp. 4957–4966 (2017)
23. Wang, F., Tong, H., Lin, C.Y.: Towards evolutionary nonnegative matrix factorization. In: AAAI, pp. 501–506 (2011)
24. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: SIGIR, pp. 165–174 (2019)
25. Wu, L., He, X., Wang, X., Zhang, K., Wang, M.: A survey on neural recommendation: from collaborative filtering to content and context enriched recommendation. arXiv preprint arXiv:2104.13030 (2021)
26. Wu, L., Li, J., Sun, P., Hong, R., Ge, Y., Wang, M.: DiffNet++: a neural influence and interest diffusion network for social recommendation. IEEE Trans. Knowl. Data Eng. (2020)
27. Wu, L., Sun, P., Fu, Y., Richang, H., Xiting, W., Meng, W.: A neural influence diffusion model for social recommendation. In: SIGIR, pp. 235–244 (2019)
28. Zhang, M., Chen, Y.: Inductive matrix completion based on graph neural networks. In: ICLR (2020)
29. Zhang, Y., et al.: How to retrain a recommender system? In: SIGIR, pp. 1479–1488 (2020)
30. Zhu, Y., et al.: Learning to warm up cold item embeddings for cold-start recommendation with meta scaling and shifting networks. arXiv preprint arXiv:2105.04790 (2021)