Fast Adaptation for Cold-start Collaborative Filtering with Meta-learning

Tianxin Wei[1], Ziwei Wu[1], Ruirui Li[2], Ziniu Hu[2], Fuli Feng[3], Xiangnan He[1], Yizhou Sun[2], Wei Wang[2]
[1]University of Science and Technology of China, China
[2]University of California, Los Angeles, USA
[3]National University of Singapore, Singapore
{rouseau, wzw9875}@mail.ustc.edu.cn, {rrli, bull, yzsun, weiwang}@cs.ucla.edu, {fulifeng93, xiangnanhe}@gmail.com,

***Abstract*—Collaborative Filtering (CF), as one of the most popular approaches, is widely employed in recommender systems but suffers from the cold-start problem, where interactions are very limited for new users in the system. To deal with this issue, previous work has largely focused on utilizing various auxiliary information such as user profiles and social relationships to infer user preferences. However, the auxiliary information is not always available due to reasons such as user privacy concerns, making the CF approaches have to count on the limited interactions. Moreover, real-world situations require both accurate and quick recommendations for newly arrived users dynamically. Therefore, it is of critical importance to enable fast learning for new users during the training time of CF models.**

In this paper, we present a novel learning paradigm, named MetaCF, to learn an accurate CF model that makes fast adaptation on new users with limited interactions. Inspired by meta-learning, MetaCF treats the fast adaptation on a new user as a task and aims to learn a suitable model for initializing the adaption. To pursue a well-generalized model, MetaCF is equipped with a *Dynamic Subgraph Sampling* that accounts for the dynamic arrival of new users by dynamically generating representative adaptation tasks for existing users. Moreover, to stabilize the adaption procedure that faces the shortage of training samples, MetaCF further optimizes the learning rates for adaption in a fine-grained manner. MetaCF is applicable to any differentiable CF-based models where we demonstrate it on two representative ones, FISM [1] and NGCF [2]. Extensive experiments on three datasets validate the effectiveness of the proposed framework, which significantly outperforms state-of-the-art baselines by a large margin in the cold-start scenario where user-item interactions are limited.

***Index Terms*—Recommendation, Meta-Learning, Cold-Start, Collaborative Filtering**

I. Introduction

Personalized recommendation has revolutionized various Web services, which largely relieves the information overload issues by matching users with appropriate items, e.g., movies on Netflix and merchandise on Amazon. Collaborative Filtering is one of the most popular approaches widely employed in recommender systems, assuming that behaviorally similar users would exhibit similar preferences on items [2]–[4]. Despite the success in serving regular users, CF approaches severely suffer from the cold-start problem, failing on new users whose interactions are very limited with inappropriate recommendations. To deal with this issue, existing CF approaches largely rely on the auxiliary information, such as user profiles [5] and social relationships [6] to facilitate the inference of user preference. However, the scalability of these approaches is limited due to the unavailability of the auxiliary information in many recommendation scenarios for reasons such as privacy concerns of users.

In this work, we argue that a good CF-based recommender system should go beyond counting on the auxiliary information to address the sparsity issue and seek techniques to improve cold-start recommendations via fast learning over the limited interactions. For instance, once the initial interactions on the items such as iPhone 11 and PS4 from a user arrived, the recommender system quickly adapts the user representations from an initial state to encode the preference on electronic products. Moreover, a good recommender system should also enable the fast adaptation of CF-based models for new users, which has received relatively little scrutiny. This is because new users arrive dynamically and the very limited new interactions can reflect user preferences dramatically, which are different from any existing ones. In this work, we pursue a well-generalized CF model with 1) generalization ability from predicting preference of existing users to new users; 2) fast and accurate adaption when fine-tuned with very limited training samples, i.e., observed interactions.

Meta-learning has empirically shown the ability to enable fast model adaption across similar tasks in standard few-shot prediction tasks such as visual recognition [7] and neural machine translation [8]. The core idea of meta-learning is learning-to-learn, i.e., learning to solve the tasks well and pursuing the generalization ability on future tasks. Technically, meta-learning [9] trains model over a large number of tasks with limited training samples in each task. It optimizes model parameters according to the adaption performance on these tasks, so the learned model can be fast adapted and generalize well on future tasks. Therefore, employing a similar approach to that of meta-learning on a CF model would also be helpful to the cold-start recommendation. That is, treating the adaption on each new user as a future task and training the CF model with similar adaption tasks on existing users.

However, directly employing existing meta-learning methods on cold-start recommendation is insufficient due to the following reasons: 1) The challenge of constructing tasks on existing users similar to the future adaption task on new users, which is critical for pursuing a model with generalization

ability. It is not wise to construct training tasks by simply extracting the historical interactions of an existing user as training samples, and treating the whole observed interactions as references to make recommendations. This is because the existing users in the system have much more interactions as training samples than new users do in the cold-start scenario. Moreover, the behavior of new users naturally different from that of regular users who are familiar with the system. 2) Performing adaption on a task via a regular model fine-tuning as conventional meta-learning methods is infeasible for cold-start recommendation. This is because of the difficulty of deciding the learning-rate suitable for updating models for different users, considering the divergence of users.

In this paper, we propose a novel meta-learning paradigm, named MetaCF, aiming to learn the well-generalized CF model. The key novelty of MetaCF is the *Dynamic Subgraph Sampling* to construct representative training tasks. To avoid over-fitting, we dynamically sample subgraph centered at a user in the training phase to account for the effect of limited interactions of new users. Furthermore, we extend the historical interactions by *Incorporating Potential Interactions* to bridge the gap between training tasks in history and future tasks and facilitate generalization. Moreover, MetaCF is equipped with *Flexible Model Updating*, which optimizes the learning-rate to perform fine-tuning according to the performance of adaption, i.e., how well the tasks are solved. It largely eliminates the tedious manual exploration of this highly sensitive hyper-parameter, and thus enables a fine-grained setting with parameter-wised learning rates which largely stabilizes the fine-tuning procedure.

The major contributions are summarized as follows:

- To the best of our knowledge, we are the first to study cold-start CF without relying on user auxiliary information. We propose a novel learning paradigm, named MetaCF, that aims to learn an accurate CF model well-generalized for fast adaptions on new users with limited interactions.
- We devise a Dynamic Subgraph Sampling strategy that constructs representative training tasks on existing users, an Incorporating Potential Interactions strategy to facilitate generalization, and a Flexible Model Updating strategy to optimize the learning rates for adaption in a fine-grained manner to stabilize fine-tuning.
- We implement MetaCF on two representative CF models, FISM and NGCF. MetaCF is further evaluated on three public real-world cold-start recommendation datasets. The results show that MetaCF significantly outperforms state-of-the-art methods.

II. Problem Definition

Let $U_e = \{u_1^e, u_2^e, ...u_n^e\}$ and $I_e = \{i_1^e, i_2^e, ...i_m^e\}$ denote the sets of existing users and items in the system respectively, where n is the number of users, and m is the number of items. The user-item interactions are represented by a bipartite graph, $G = (U_e \cup I_e \cup Y^e)$, where each vertex represents a user or an item. $Y^e \in \mathbb{R}^{n \times m}$ denotes the existence of interactions between users and items. Formally,

$$Y_{ui}^e = \begin{cases} 1, & \text{if user } u \text{ has interacted with item } i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The setting of regular recommendation is to learn a scoring function $f(u, i|\theta)$ from G to predict the preference of an user u over item i, which is largely focused on the existing users U_e. θ denotes the parameters to be learned. Cold-start recommendation is focused on new users arriving after the training stage. Given a set of new users $U_f = \{u_1^f, u_2^f, ...u_k^f\}$, the objective is to quickly adapt the function $f(u, i|\theta)$ learned from G to be a user specific one $f(u, i|\theta_u)$ according to user's initial interactions D_{u^f} where $D_{u^f} = \{i|u^f$ has interacted with item $i\}$, so as to properly predict the preference of new user. The size of user's initial interactions is assumed to be smaller than K which is set as a small number (e.g., 1, 2 or 3) in consideration of the cold-start scenario[1]. Formally, we treat the fast adaption on each new user as a task:

$$\tau_u : (\theta, D_{u^f}) \rightarrow \theta_{u^f}, \tag{2}$$

which we denote as τ_u. A straightforward solution is to learn θ from G as regular recommendation, and fine-tune θ over D_{u^f}. However, it is easy to encounter the over-fitting issue, since there are very few interactions available for new users. In this work, we aim to learn a proper θ from G that generalizes well on new users, i.e., solving the fast adaption tasks. Considering that the learning target is how to solve the tasks well (i.e., with a good generalization ability on future tasks), addressing this problem can be seen as an instance of meta-learning. Note that even though we argue the unavailability of auxiliary information in some recommendation scenarios, the information can be easily integrated into our framework as an additional input of each task τ_u (see Section 4.2 for detailed results).

III. Proposed Method

In this section, we present the details of MetaCF. As shown in Figure 1, the framework of MetaCF is an instance of meta-learning, which aims to learn a meta-model (can be any differentiable CF model) well-generalized for solving the fast model adaption task on a new user. We first detail the meta-learning paradigm in Section 3.1, followed by the description of a novel dynamic subgraph sampling method to construct representative training tasks on existing users from historical interactions with the consideration of potential future interactions (Section 3.2). Lastly, we present the instantiation of MetaCF implemented on two representative CF models, FISM and NGCF.

A. Meta-learning for Fast Adaption

Meta-learning has shown remarkable success in achieving fast model adaption across similar tasks in various few-shot prediction tasks such as visual recognition [7] and neural

[1]For the new users without any observed interactions, we can recommend the most trendy items in the system to collect their initial interactions.

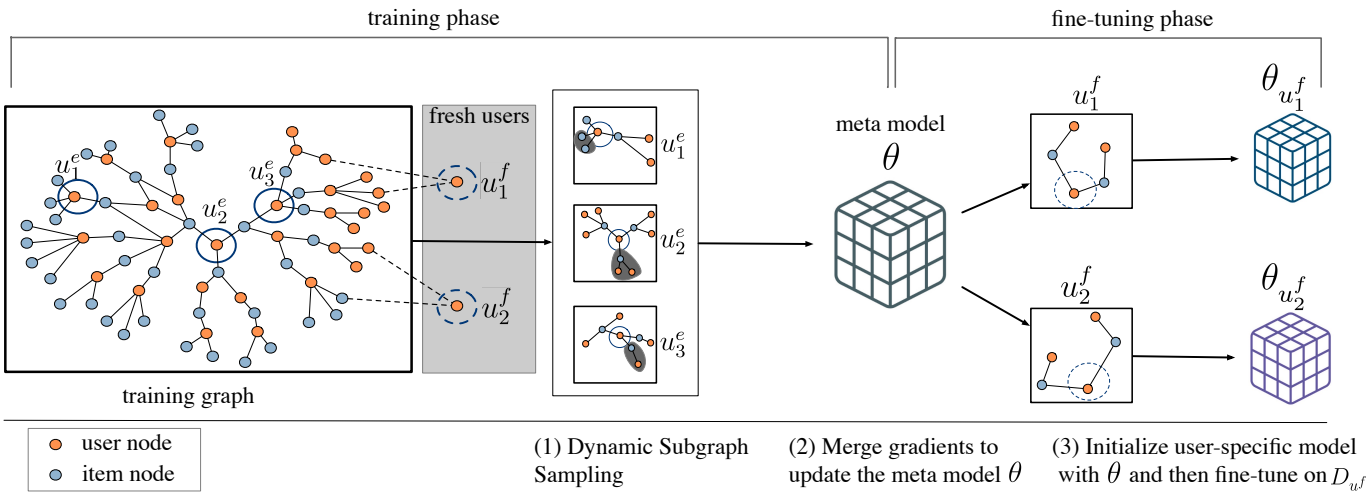


Fig. 1. The framework of MetaCF. The black rectangle border denotes the historical user-item interactions. The gray rectangle denotes the new users, who are unseen in the training phase. The orange and blue nodes stand for users and items, respectively. (1) We randomly sample a collection of existing users and subgraphs centered at each user to construct tasks. (2) We perform fine-tuning on each task and calculate the gradient to update the meta-model according to performance on the task. The meta-model is then updated by aggregating the gradients from all tasks. (3) For a new user when testing, we fine-tune the meta-model to generate a personalized model for the user.

machine translation [8]. The core idea of meta-learning is learning-to-learn, i.e., learning to solve the tasks well and pursuing the generalization ability on future tasks. For instance, in the visual recognition task, meta-learning aims to learn an image classification model that can be quickly adapted to classify images from unseen classes with a few training samples. Similarly, employing a similar approach to that of meta-learning on a CF model would also be helpful to the cold-start recommendation, i.e., solving the fast adaption task τ_u on new users. We follow one of the popular meta-learning frameworks, MAML [9], for the consideration of its model agnostic property to avoid the overhead of adjustment for the specific CF model. To learn the parameters of a new task, MAML trains the model in such a manner that the parameters of the new task will have a promising initialization by learning from various similar tasks. From the learned parameters, the model is further fine-tuned on the new task with only a few interactions over several shots of training samples for the new task.

Similarly, to achieve fast adaption on the future tasks τ_u on new users, the proposed MetaCF iteratively updates the meta-model according to the following procedure:

- **Inner loop.** For each task τ_u, MetaCF initializes θ_u with the latest parameters of the meta-model θ, and updates θ_u according to the user's training interactions.
- **Outer loop.** According to the adaption performance on each task τ_u, i.e., the recommendation loss of θ_u regarding user u, MetaCF updates the meta-model θ.

To be more specific, in each iteration/batch, a fixed number of N tasks are included, where each task involves the parameter optimizations of one particular user. More precisely, we first sample a collection of N users $B = \{u_n\}_{n=1}^{N}$ from U_e. Then, for each user in the collection, we sample subgraphs between the user and corresponding positive and negative items based on the algorithms in Section III-B. Note that the number of sampled interactions per user in each task is upper bounded by K. In particular, for each episode, we randomly sample $k \in [1, K]$ interactions of the user for training. This increases the flexibility of the model as it allows us to choose a wide range of interactions as references to make recommendations for a user. This strategy is important, as users have different numbers of interactions to refer to during testing, and a fixed k does not work for all the users. The whole algorithm is summarized in Algorithm 1. As can be seen, in the test stage, the new user $u's$ specific model parameter θ_u is initialized with the final θ and then fine-tuned based on his/her limited observed interactions in D_{u^f}. In the following, we describe details of the inner loop and outer loop.

Inner loop. The model parameters regarding a user u are updated iteratively T_{train} times as follows:

$$\begin{aligned} \theta_u^{(t)} &= \theta_u^{(t-1)} - \alpha \nabla_{\theta_u^{(t-1)}} J_s(\theta_u^{(t-1)}, D_u^s, G_u^s), \\ J_s(\theta_u^{(t-1)}, D_u^s, G_u^s) &= J(\theta_u^{(t-1)}, D_u^s, G_u^s) + \lambda \|\theta_u^{(t-1)}\|^2, \end{aligned} \tag{3}$$

where J denotes the recommendation loss (e.g., the log loss [10] or pairwise loss [11]) on data D_u^s with θ_u as the recommender model parameters. D_u^s, G_u^s are user's sampled data and subgraphs respectively according to Section III-B. T_{train} is a predefined hyper-parameter controlling the number of training updates, $\theta_u^{(t)}$ is the model parameters of user u at time step t and $\theta_u^{(0)}$ is initialized with θ. α is the learning rate of user parameter update. λ is a hyper-parameter to balance the loss and the regularization term. Note that the model parameters of user u at time step 0 are initialized by the generalized model parameters θ.

Outer loop. We then update the generalized model parameters by summing up all user u's specific loss $J_u(\theta_u^{(T_{train})})$ in this collection together. More concretely, in each batch, the

parameters are updated as follows:

$$\theta = \theta - \beta \nabla_\theta \sum_{u_i \in B} J(\theta_{u_i}^{(T_{train})}, D_{u_i}^q, G_{u_i}^q) \tag{4}$$

where β is learning rate of the generalized model parameters, $D_{u_i}^q$ and $G_{u_i}^q$ are another sampled dataset and subgraphs according to Section III-B which are different from $D_{u_i}^s$ and $G_{u_i}^s$, B is a set of users involved in a batch, $\nabla_\theta \sum_{u_i \in B} J_{u_i}(\theta_{u_i}^{(T)}, D_{u_i}^q)$ is the meta gradients and θ is updated iteratively until convergence.

Testing. As shown in Algorithm 1 (line 18-24), for evaluation of new user u_k^f, we first initialize the user model parameter with θ, and the user model is then fine-tuned T_{test} times with his/her observed interactions where T_{test} is the number of update for fine-tuning. The fine-tuned model is then applied to make recommendations.

1) Flexible Model Updating: Similar to the discovery in [12], we also find that methods utilizing MAML are very sensitive to the setting of the user parameter learning rates α, that is, a subtle change of the learning rate α can lead to a dramatic performance drop. The training data of those users who have few interactions are of very small size. It will lead to over-fitting and unstable performance when only a small dataset is available [13], [14]. A manually fixed learning rate can even make the model unable to converge since it is hard to define a proper one for a small dataset.

To learn to set the appropriate learning rates automatically, we implement a flexible update strategy. We highlight that this is different from conventional MAML-based methods, where learning rates have to be hand-crafted and heavy tuning efforts are involved. The proposed model has a higher capacity by learning to learn not only the learner's initialization but also the learner's learning rate.

In particular, for each trainable parameter matrix $w \in \mathcal{R}^{AxB}$, we assign α as a vector of the same size as w that decides both the update direction and learning rate as follows:

$$w_u^{(t)} = w_u^{(t-1)} - \alpha \nabla_{w_u^{(t-1)}} J_s(w_u^{(t-1)}, D_u^s, G_u^s)$$

$$w = w - \beta \nabla_w \sum_{u_i \in B} J_q(w_{u_i}^{(T_{train})}, D_{u_i}^q, G_{u_i}^q)$$

$$\alpha = \alpha - \beta \nabla_\alpha \sum_{u_i \in B} J_q(w_{u_i}^{(T_{train})}, D_{u_i}^q, G_{u_i}^q)$$

where D_u^s, G_u^s and D_u^s, G_u^s are all sampled according to Algorithm 2. $w_u^{(0)}$ is initialized with w. line 14 in Algorithm 1 gives the details of the above update for user parameter learning rates.

B. Fast Adaptation Task Generation

1) Dynamic Subgraph Sampling: The first step of our framework is to sample subgraphs of the interested user u and k interactive items out of the interaction graph defined in Section II in each task τ_u. Taking in the same k interactions as references in both training and testing is impractical. It is difficult to choose an appropriate k since different users

Algorithm 1 Personalized Fast Cold-Start User Adaption

Input: training user distribution $p(u)$, testing new user U_f, training update size T_{train}, fine-tune update size T_{test}, sample ceiling K
Initialize: θ, β, α

1: /* Training on the existing users */
2: **while** not converge **do**
3: sample batch of users $B \sim p(u)$;
4: **for** user u_i in B **do**
5: $\theta_{u_i}^{(0)} = \theta$;
6: **for** t in range(T_{train}) **do**
7: $D_{u_i}^s$, $G_{u_i}^s$ = Dynamic Subgraph Sampling(u_i, K) ;
8: $\theta_{u_i}^{(t)} = \theta_{u_i}^{(t-1)} - \alpha \nabla_{\theta_{u_i}^{(t-1)}} J_s(\theta_{u_i}^{(t-1)}, D_{u_i}^s, G_{u_i}^s)$
9: **end for**
10: $D_{u_i}^q$, $G_{u_i}^q$ = Dynamic Subgraph Sampling(u_i, k);
11: Evaluate $J(\theta_{u_i}^{(T_{train})}, D_{u_i}^q, G_{u_i}^q)$
12: **end for**
13: $\theta = \theta - \beta \nabla_\theta \sum_{u_i \in B} J(\theta_{u_i}^{(T_{train})}, D_{u_i}^q, G_{u_i}^q)$
14: $\alpha = \alpha - \beta \nabla_\alpha \sum_{u_i \in B} J(\theta_{u_i}^{(T_{train})}, D_{u_i}^q, G_{u_i}^q)$
15: **end while**
16:
17: /* Testing on new users */
18: **for** new user j in U_f **do**
19: Use all observed interactions to generate $D_{u_j}^s$, $G_{u_j}^s$, to fine-tune;
20: $\theta_{u_j}^{(0)} = \theta$;
21: **for** t in range(T_{test}) **do**
22: Compute adapted parameters $\theta_{u_j}^{(t)}$ with gradient descent by $\theta_{u_j}^{(t)} = \theta_{u_j}^{(t-1)} - \alpha \nabla_{\theta_{u_j}^{(t-1)}} J_s(\theta_{u_j}^{(t-1)}, D_{u_j}^s, G_{u_j}^s)$
23: **end for**
24: Recommend unobserved items to user j base on $\theta_{u_j}^{(T_{test})}$
25: **end for**

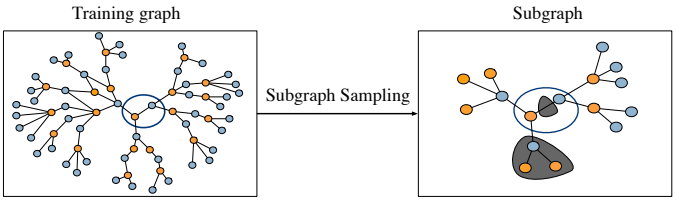


Fig. 2. The proposed method of subgraph sampling between a user-item pair. The blue circle represents the selected user-item pairs. The removed interactions, shown in the shadow part, are not used for the current episode. The yellow and orange nodes stand for users and items.

involve different numbers of interactions in practice and a fixed k does not work well for all users. To improve the flexibility and generalization of the model, we design a dynamic K-shot training mechanism by constructing each training batch of users with a dynamic number of interactions as shown in Figure 2. To mimic the test setting of cold-start scenarios, as shown in Algorithm 2, we randomly sample interactions per user that are used for training and hide the remaining interactions, which serve as interactions to be predicted for validation. In particular, the number of sampled interactions per user in each task is upper bounded by K. For each update, we randomly sample $k \in [1, K]$ interactions for training. This allows us to choose a wide range of interactions as references to make recommendations for a user during testing based on how many interactions are available. In this way, MetaCF not only learns from different combinations of

user-item interactions, but also becomes more flexible when conducting recommendations.

Algorithm 2 Dynamic Subgraph Sampling

Input: the bipartite graph G, the target user u, hop number h, sample ceiling K
Output: sampled data D_u,sampled subgraphs G_u

Choose k $\in$ [1, K] by random.
Randomly sample k edges D_u^n not connected to u in G as negative samples.
Randomly sample k edges D_u^p connected to u in G as positive samples and remove the remaining edges.
G_u = {}
D_u = $D_u^n \cup D_u^p$
for each item i in $D_u^n \cup D_u^p$ **do**
  Remove the edge between user u and item i if exists
  U = U_{fringe}= { u }, I = I_{fringe}= { i }
  for j = 1,2,...,h **do**
    U'_{fringe}= { u_j: $u_j \sim I_{fringe}$} \ U
    I'_{fringe}= { i_j: $i_j \sim U_{fringe}$} \ I
    U_{fringe}= U'_{fringe}, I_{fringe}= I'_{fringe}
    U = U $\cup$ U_{fringe}, I = I $\cup$ I_{fringe}
  end for
  $G_{u,i}$is the vertex-included subgraph from G using vertices U,I
  G_u = $G_u \cup G_{u,i}$
end for

Note: $\{u_j$: $u_j \sim I_{fringe}\}$ is the set of nodes that are adjacent to at least one node in I_{fringe}

2) Incorporating Potential Interactions: Considering that there would be a gap between the tasks generated purely on the historical interactions and future tasks. We further incorporate the potential interactions from existing users to bridge the gap by mining the structure of the interaction graph to infer the indirect relations between users and items. The indirect relations between the items in the graph are preserved and presented in the latent space in unsupervised graph embedding models. Specifically, we develop a novel module by incorporating potential interactions based on the network embedding, which allows the model to capture long-range dependencies and increase generalization ability to the user-item pair that are not connected in the graph. Here we follow BiNE [15], a random walk based network representation model, to construct a bipartite graph and derive embeddings Z for items. The structural similarity between item i and item j is defined as:

$$d(z_i, z_j) = -||z_i - z_j||^2,$$

where z_i is the embedding for item i, $d(z_i, z_j)$ is the latent similarity between item i and item j. The interest of user u to item j is defined as:

$$I(u, j) = \sum_{i \in N(u)} d(z_i, z_j).$$

Different from ItemCF, the proposed model measures the similarity of items comprehensively utilizing the structural information in the entire bipartite graph, which not only captures the neighborhood information but also leverages long-range implicit relationships. We highlight that it is important for the embedding learning, especially beneficial to alleviate the cold-start problem. To be specific, for each user u, we rank the interest of all items that are not connected to u and select the highest k items to add to the neighboring nodes, where k is the same as the number of dynamic sampling. All the neighboring nodes are used to conduct dynamic subgraph sampling. The newly incorporated links are only used to create the subgraph and will not be used as the positive and negative samples.

C. Model Instantiation

To demonstrate how the proposed MetaCF works, we implement it on two representative CF-based models, FISM [1] and NGCF [2]. Both FISM and NGCF make predictions as:

$$f(u, i|\Theta) = (e_u)^T e_i, \tag{5}$$

where e_u and e_i denote the embedding of user u and item i. The differences between FISM and NGCF are the way to calculate the embedding and the according learnable parameters.

FISM. As an item-based CF model, FISM treats embedding of items as learnable parameters and calculates user embeddings from the interacted items. Formally,

$$e_u = \frac{1}{|N(u)|^\gamma} \sum_{j \in N(u)} e_j, \tag{6}$$

where γ is a hyper-parameter controlling the normalization effect; $N(u)$ denotes the items interacted by user u. For positive user-item pairs, the target item i is removed from $N(u)$ to avoid the modeling of self-similarity of the target item. In training phase, MetaCF optimizes all the learnable parameters of FISM, i.e., $\Theta = \{e_i | i \in [1, m]\}$.

NGCF. NGCF performs graph convolution over the subgraph $G_{u,i}$ centered at user u and item i to obtain user and item representations, which propagate the embedding of user and item over $G_{u,i}$ to account for the global high-order interdependencies among users and items. We employ a d-layer NGCF where the embedding of item i is iteratively propagated to its d-hop neighbors. Different from the standard NGCF where the embedding of both users and items are model parameters, we devise a variant of NGCF called Inductive NGCF which omits the parameters on user embedding and initiate the embedding propagation from item to user. Algorithm 3 shows the details of the propagation procedure where σ denotes an activation function; $W_I^{(l)}$ and $W_U^{(l)}$ are model parameters at layer l performing embedding transformation. In the training stage of Inductive NGCF, MetaCF learns the parameters of both item embedding and transformation, i.e., $\Theta = \{e_i | i \in [1, m]\}, \{(W_I^{(l)}, W_U^{(l)}) | l \in [0, d]\}$.

Loss Function. For both FISM and NGCF, we adopt the widely used pairwise Bayesian Personalized Ranking (BPR) [11] as the recommendation loss J. For each observed positive user-item interaction $(u, i) \in D_u$, we randomly sampling a negative interaction (u, j) as described in Algorithm 2. The BPR loss is formulated as:

$$J(\theta, D_u) = \sum_{(u,i) \in D_u} -ln\sigma(f(u, i|\theta) - f(u, j|\theta)), \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function.

IV. Experiments

To justify the effectiveness of MetaCF, we conduct extensive experiments to answer the following research questions:

Algorithm 3 Inductive NGCF

Input: Subgraph $G_{u,i}$, User u, Item i, NGCF depth d, Activation function σ
Output: e_u, e_i

for l in range(d) **do**
  $m_{u\leftarrow i}^{(l)} = \frac{W_I^{(l)} e_i^{(l-1)}}{\sqrt{|N_u||N_i|}}$ /*$N(u), N(i)$ denote the neighbors of u, i*/
  $e_u^{(l)} = \sigma(m_{u\leftarrow u}^{(l)} + \sum_{j\in N(u)} m_{u\leftarrow j}^{(l)})$
  $m_{i\leftarrow u}^{(l)} = \frac{W_U^{(l)} e_u^{(l-1)}}{\sqrt{|N_u||N_i|}}$
  $e_i^{(l)} = \sigma(m_{i\leftarrow i}^{(l)} + \sum_{j\in N(i)} m_{i\leftarrow j}^{(l)})$

end for
$e_u = e_u^{(l)}, e_i = e_i^{(l)}$

- **RQ1**: How does MetaCF perform compared with state-of-the-art item recommendation methods?
- **RQ2**: How do different modules of our MetaCF (e.g., the dynamic subgraph sampling module) contribute to the performance?
- **RQ3**: What is the effect of our MetaCF on users with different levels of sparsity?
- **RQ4**: How do different hyper-parameters settings affect MetaCF?

A. Experiment Settings

TABLE I
STATISTICS OF THE DATASETS.

Dataset	Users	Items	Interactions	Density
Amazon-Electronics	192,403	63,001	1,689,188	0.014%
Amazon-Kindle	68,223	61,885	982,619	0.023%
Last-FM	1,892	1,7632	92,834	0.278%

To evaluate the effectiveness of our approach, we conducted experiments on three widely used benchmark datasets in different domains. The first two datasets are from the Amazon collection [16], which contains product reviews from Amazon ranging from May 1996 to July 2014. We follow [17] and select Amazon-Electronics and Amazon-Kindle as the first and the second dataset. The third dataset is collected from Last-FM, which is an official song tag dataset and the artists are viewed as the items. We follow [2] and use the 10-core setting for these datasets to ensure that each user and item has at least ten interactions. All the datasets above are publicly available.

We conduct inductive personalized recommendation experiments on these three datasets. For each dataset, 10% of the users are involved in the validation set and another 10% of the users are involved in the testing set. These users are excluded from the construction of the training graph. Next, for each test and validation user, represented by a node in the graph, we select a portion of the interactions as the observed edge in the validation and test phase, i.e. the positive samples. The remaining interactions in the validation and test set serve as the validation and test data. That is, the trained model has never seen this part of the nodes in the training, so new items need to be recommended based on the edges the user observed. We organize the experiment with three different sizes of observation edges, namely 5-shot, 3-shot and 1-shot, which means that each new node has 5, 3 and 1 edges that can be used as the input for model validation and test. To evaluate the performance of the item recommendation, we adopt the leave-one-out evaluation, which has been widely used [1], [4], [18]. More precisely, for each user in the validation and testing set, we leave his latest interaction for validation and test. For each user node in the test and validation set, we take each observed edge as a positive sample of the user, and then randomly select 100 items that did not interact with the current user as the negative samples. This method has been widely used in many other works [4], [18]–[21]. Then we rank the list consisting of the positive item and 100 negative items. Following [2], we use Hit Ratio at rank 10 (HR@10) and Normalized Discounted Cumulative Gain at rank 10 (NDCG@10) as the evaluation metrics to measure the ranking performance.

1) Baselines: We compare MetaCF with the following baselines:

- **DropoutNet [22]:** DropoutNet combines the dropout technique with a deep neural network to learn effective features of the input to solve the cold-start problem. To make it adapted to our setting, we treat interactions as features.
- **FISM [1]:** This is a representative item-based CF model as formulated in Equation (4).
- **NGCF [2]:** We use the inductive version of Neural Graph Collaborative Filtering, using the subgraph sampling strategy in the training process.
- **MeLU [23]:** MeLU presents a meta-learning strategy to address the user cold-start problem. As we focus on modeling user behaviors, to make a fair comparison, we use the user/item historical interactions as the user/item features in this model.

Note that we omit the comparison with potential baselines of streaming recommendation [24] since their objective is to perform fast model adaption for all users instead of personalized adaption for specific cold-start users.

2) Parameter Settings: In the training stage, we set the dimensions of embedding vectors of all the above models to 64 by default. For the NGCF model, we utilize two layers of embedding propagation network to obtain the best results. For the FISM model, We test γ from 0 to 1 with a step size of 0.1, finding a value of 0 leads to the best results on all datasets. For a fair comparison, for all models, we optimize them with bayesian personalized ranking [11] loss via the mini-batch Adam [25] algorithm. We set the number of training epochs to 100 and the number of batches in each epoch to 512. We train 64 tasks in a batch and set the upper bound of k-shot to 5 in MetaCF. The inner learning rate is initialized to 1e-3 and then optimized by our algorithm. The training update step and fine-tuning update step are both set to 4. Without special mention, we report the performance of MetaCF with following default settings: learning rate=1e-3, dropout rate=0.2 [26], weight decay=1e-4.

B. Results (RQ1)

Table II shows the recommendation performance of all compared methods w.r.t. HR@10, and NDCG@10 on the Amazon-Electronics, Last-FM, and Amazon-Kindle datasets

TABLE II
EXPERIMENTAL RESULTS OF DIFFERENT METHODS OVER THE THREE DATASETS.

Model			DropoutNet	MeLU	FISM	MetaCF$_{FISM}$	NGCF	MetaCF$_{NGCF}$
Electronics	HR@10	1-shot	0.111	0.246	0.201	0.310	0.285	**0.401**
		3-shot	0.214	0.288	0.263	0.325	0.335	**0.427**
		5-shot	0.246	0.345	0.317	0.391	0.387	**0.434**
	NDCG@10	1-shot	0.066	0.127	0.115	0.177	0.147	**0.239**
		3-shot	0.112	0.167	0.157	0.246	0.182	**0.254**
		5-shot	0.134	0.202	0.189	0.273	0.214	**0.257**
LastFM	HR@10	1-shot	0.284	0.413	0.394	0.443	0.463	**0.508**
		3-shot	0.448	0.466	0.414	0.488	0.519	**0.534**
		5-shot	0.477	0.489	0.442	0.498	0.528	**0.547**
	NDCG@10	1-shot	0.208	0.251	0.229	0.289	0.266	**0.304**
		3-shot	0.274	0.238	0.227	0.297	0.312	**0.325**
		5-shot	0.291	0.255	0.237	0.298	0.317	**0.331**
Kindle	HR@10	1-shot	0.136	0.389	0.375	0.403	0.410	**0.512**
		3-shot	0.328	0.512	0.508	0.539	0.530	**0.626**
		5-shot	0.440	0.531	0.513	0.542	0.567	**0.656**
	NDCG@10	1-shot	0.117	0.222	0.224	0.276	0.219	**0.295**
		3-shot	0.245	0.291	0.286	0.319	0.295	**0.382**
		5-shot	0.299	0.318	0.315	0.337	0.321	**0.407**

under the 1/3/5-shot settings. From the table, we have the following findings:

- We observe that MeLU, where the meta-learning strategy is used for new users, has a much better performance than DropoutNet. This proves the advantage of meta-learning in the cold-start problem in recommender systems.
- Without MetaCF, NGCF can make a relatively more accurate recommendation for new users than other baselines. This indicates the significance of using high-order interactions, or structural information when few interactions are available for new users.
- We notice that MetaCF outperforms all baseline methods under all settings. This indicates the importance of both reasonably utilizing the local and global information and fast adaptation on new users in the cold-start scenario in recommender systems.
- By leveraging our framework on collaborative filtering models, we improve their performance significantly on the three datasets. For instance, compared with NGCF, MetaCF improves the HR@10 performance from 0.285, 0.335, 0.387 to 0.401, 0.427, and 0.434 for the 1/3/5-shot setting on the Amazon-Electronics dataset, respectively. This demonstrates the effectiveness of fast adaptation utilizing our framework on top of CF-based models.

Incorporating Side Information: Our approach can not only leverage local and global information but also well utilize side information. To justify this, additional experiments were conducted on the Amazon-Electronics dataset. In particular, user and item reviews are used to serve as side information. To represent reviews, we first leverage GloVe [27] to obtain the word embeddings. The review embedding is further derived by taking the average of involved word embeddings in the review. The constructed review embeddings are further integrated into different methods to serve as side information. The corresponding results are summarized in Table III. The proposed method with consideration of side information achieves further improvement and consistently outperforms MeLU.

TABLE III
EFFECT OF SIDE INFORMATION.

Model	Electronics		LastFM		Kindle	
	HR	NDCG	HR	NDCG	HR	NDCG
MeLU	0.246	0.127	0.413	0.251	0.389	0.222
MeLU+SI	0.301	0.156	0.439	0.277	0.398	0.262
MetaCF$_{NGCF}$	0.401	0.239	0.508	0.304	0.512	0.295
MetaCF$_{NGCF}$+SI	**0.423**	**0.254**	**0.533**	**0.331**	**0.539**	**0.308**
MetaCF$_{FISM}$	0.310	0.177	0.443	0.289	0.403	0.276
MetaCF$_{FISM}$+SI	0.331	0.198	0.456	0.303	0.428	0.289

C. In-depth Analysis of MetaCF (RQ2)

We then investigate the effectiveness of the proposed MetaCF under different circumstances.

Recall that MetaCF has three essential components: dynamic subgraph sampling, incorporating potential global interactions, and flexible model update strategy. Different MetaCF variants are designed for comparison of its components. We summarize them as follows:

- **MetaCF**$^{\neg D}$: A variant of MetaCF with the dynamic subgraph sampling module being removed.
- **MetaCF**$^{\neg G}$: A variant of MetaCF with the global long-range interaction being removed, using only the existing interactions.
- **MetaCF**$^{\neg F}$: A variant of MetaCF with the flexible model update strategy being removed.

Different models are summarized in Table IV, where the symbol ✓ indicates the algorithm exploits the corresponding information. The Local means that the model aggregates the neighborhood information in the graph. We use Adapt to represent the ability to adapt to new users. The DGS means the dynamic subgraph sampling module. The Global represents the model with potential long-range interactions. The FMU is the flexible model update strategy for meta-learning.

1) Effect of Dynamic Subgraph Sampling: We first explore the impact of dynamic graph sampling (DGS) that uses up to k-shot neighborhood information of each user. Table V summarizes the experimental results obtained under 1-shot

TABLE IV
THE SUMMARY OF DIFFERENT MODELS.

Methods	Local	Adapt	DGS	Global	FMU
DropoutNet					
FISM	✓				
NGCF	✓				
MeLU		✓			
MetaCF$^{\neg D}$	✓	✓		✓	✓
MetaCF$^{\neg G}$	✓	✓	✓		✓
MetaCF$^{\neg F}$	✓	✓	✓	✓	
MetaCF	✓	✓	✓	✓	✓

setting. We note that the MetaCF with dynamic subgraph sampling outperforms MetaCF$^{\neg D}$ on all the datasets as it imitates the few-shot setting in the testing stage. The same trend can be observed about FISM.

TABLE V
EFFECT OF DYNAMIC SUBGRAPH SAMPLING.

Model	Electronics		LastFM		Kindle	
	HR	NDCG	HR	NDCG	HR	NDCG
MetaCF$^{\neg D}_{NGCF}$	0.348	0.197	0.442	0.256	0.435	0.245
MetaCF$_{NGCF}$	**0.401**	**0.239**	**0.508**	**0.304**	**0.512**	**0.295**
MetaCF$^{\neg D}_{FISM}$	0.267	0.145	0.417	0.248	0.355	0.231
MetaCF$_{FISM}$	**0.310**	**0.177**	**0.443**	**0.289**	**0.403**	**0.276**

2) Effect of Incorporating Potential Interactions: To explore the effect of incorporating potential interactions that utilize the structural information to do data augmentation for users, we use MetaCF$^{\neg G}_{NGCF}$ to indicate the MetaCF model without the module of incorporating potential interactions. Table VI summarizes the experimental results which are obtained on the Amazon-Electronics dataset o NGCF model. We note that the MetaCF with the module of incorporating potential interactions outperforms MetaCF$^{\neg G}$, especially in 1-shot setting. It is clear that our method can alleviate data sparsity to some extent at a small cost.

TABLE VI
EFFECT OF INCORPORATING POTENTIAL INTERACTIONS.

	MetaCF$_{NGCF}$		MetaCF$^{\neg G}_{NGCF}$	
	HR	NDCG	HR	NDCG
1-shot	**0.401**	**0.239**	0.389	0.227
3-shot	**0.427**	**0.254**	0.421	0.245
5-shot	**0.434**	**0.257**	0.432	0.252

3) Effect of Flexible Model Update Strategy: To explore the impact of different model update strategies, we consider the variants of NGCF and FISM that use different settings. Our MetaCF framework dynamically learns the inner learning rate which is a very important hyper-parameter. This saves a lot of time adjusting this parameter. We compare our method with the original approach that fixes the inner learning rate. For the fixed inner learning rate, we test the values of [1e-2; 1e-3; 1e-4; 1e-5] and choose the best one according to the validation performance. We use MetaCF$^{\neg F}_{NGCF}$ to indicate leveraging MetaCF on NGCF with fixed learning rates and NGCF$_{finetune}$ to represent the NGCF model with the fine-tuning stage. Notations for FISM are similar. Table VII summarizes the experimental results under 1-shot setting. We have the following findings:

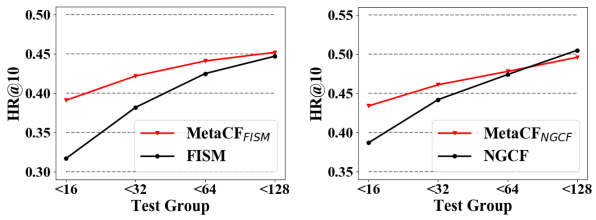
- Leveraging MetaCF framework on NGCF or FISM with the fixed inner learning rate has much worse performance than learning it dynamically, even though our framework still has made a little improvement on the initial one. We can conclude that the inner learning rate is a significant factor in our task.
- We evaluate the effect of MetaCF by comparing it with just fine-tuning. The results show that directly fine-tuning on new users can lead to extremely bad performance due to the insufficiency of data. On the contrary, adapting with MetaCF can leverage the existing users' information as regularization to avoid over-fitting.

TABLE VII
EFFECT OF FLEXIBLE MODEL UPDATE STRATEGY.

Model	Electronics		LastFM		Kindle	
	HR	NDCG	HR	NDCG	HR	NDCG
NGCF	0.285	0.147	0.463	0.266	0.410	0.219
NGCF$_{finetune}$	0.237	0.116	0.422	0.232	0.332	0.146
MetaCF$^{\neg F}_{NGCF}$	0.319	0.166	0.471	0.273	0.436	0.238
MetaCF$_{NGCF}$	**0.401**	**0.239**	**0.508**	**0.304**	**0.512**	**0.295**
FISM	0.201	0.115	0.394	0.229	0.375	0.224
FISM$_{finetune}$	0.188	0.106	0.390	0.218	0.358	0.211
MetaCF$^{\neg F}_{FISM}$	0.223	0.132	0.413	0.245	0.382	0.254
MetaCF$_{FISM}$	**0.310**	**0.177**	**0.443**	**0.289**	**0.403**	**0.276**

D. Model Sensitivity

1) Performance w.r.t. Interaction Sparsity Levels (RQ3): The sparsity issue usually limits the performance of recommender systems as we mentioned before and our framework dedicates to solving the data sparsity problem. To investigate the effect of user interactions sparsity on our framework, we conduct experiments over different sparsity levels of test user interactions. To be more specific, we divide the test users into different groups based on the number of interactions per user. For each test user, we fix the target interactions to predict in different settings of the sparsity level. Fig 3 shows how our framework's performance varies in different test groups by fixing the 5-shot training mechanism and its comparison with original FISM and NGCF on the Amazon-Electronics dataset. We have similar observations on the other two datasets.



(a) FISM (b) NGCF

Fig. 3. HR@10 performance comparison over different sparsity levels of test user groups on Amazon-electronics.

2) Impacts of Hyper-parameters (RQ4): Our MetaCF method introduces two additional hyper-parameters, i.e., training update step and fine-tuning update step, to control the

TABLE VIII
EFFECT OF TRAINING UPDATE STEPS.

	HR@10	NDCG@10
MetaCF$_{TU}$-1	0.472	0.257
MetaCF$_{TU}$-2	0.491	0.286
MetaCF$_{TU}$-3	0.505	0.291
MetaCF$_{TU}$-4	**0.512**	**0.296**
MetaCF$_{TU}$-5	0.510	0.294

number of updates for a training task and the number of fine-tuning for new users, respectively. Here we show how these two hyper-parameters impact the performance and also shed light on how to set them. All the results are tested on the Amazon-Electronics dataset with MetaCF$_{NGCF}$.

First, we fix fine-tuning update step to 4 and vary the training update steps. In particular, we search the training update step in the range of $\{1, 2, 3, 4, 5\}$. Table VIII summarizes the experimental results, wherein MetaCF$_{TU}$-4 indicates the NGCF model updated four times, and similar notations for others. As we can see, MetaCF$_{TU}$-2, 3, 4, substantially gain improvement over MetaCF$_{TU}$-1 since insufficient training updates will lead to under-fitting problem. MetaCF$_{TU}$-5 performs a little worse than MetaCF$_{TU}$-4 probably due to the inconsistency between the fine-tuning and training update step.

Then, to investigate the initial performance of MetaCF and the benefit of fine-tuning, we fix the training update step to 4 and vary the fine-tuning update step. In particular, we search the fine-tuning update step in the range of $\{0, 1, 2, 3, 4, 5\}$. Table IX summarizes the experimental results, wherein MetaCF$_{FU}$-4 indicates the model fine-tuned four times, and similar notations for others. According to the table, we draw the following conclusions:

- We observe that MetaCF$_{FU}$-0 has consistently outperformed other baselines without fine-tuning. It demonstrates the effectiveness of MetaCF$_{FU}$, indicating that the modeling of personal inference could greatly avoid over-fitting.
- MetaCF$_{FU}$-2, 3, 4 substantially gain improvement over MetaCF$_{FU}$-1. MetaCF$_{FU}$-5 performs a little worse than MetaCF$_{FU}$-4. The reason may be similar to the discussion about the training update steps.

TABLE IX
EFFECT OF FINE-TUNING UPDATE STEPS.

	HR@10	NDCG@10
MetaCF$_{FU}$-0	0.502	0.287
MetaCF$_{FU}$-1	0.503	0.290
MetaCF$_{FU}$-2	0.505	0.291
MetaCF$_{FU}$-3	0.510	0.295
MetaCF$_{FU}$-4	**0.512**	**0.296**
MetaCF$_{FU}$-5	0.511	0.295

V. RELATED WORK

In this section, we go over the related works on collaborative filtering, cold-start recommendation and meta-learning.

A. Collaborative Filtering

Due to the abundance of user feedback such as ratings and purchases that can directly reflect a user's preference, research on item recommendation has mainly focused on mining the feedback data, known as collaborative filtering (CF). Matrix Factorization (MF) [28] plays a dominant role in recommender systems. The basic principle behind MF is that we could project users or items into a latent space so that users' preferences can be reflected using their proximity to the items. Kabbur et al. [1] propose a factored item similarity model (FISM), which represents an item as an embedding vector and models the similarity between two items as the inner product of their embedding vectors. FISM provides advanced recommendation accuracy and is well suited for online recommendation scenarios. Recently deep learning has been leveraged to develop non-linear neural network models for CF. For instance, He et al. [3] employs nonlinear neural networks as the interaction function. Graph neural network models for CF [2], [29]–[31] have been developed utilizing deep learning and graph learning. Wang et al. [2] exploit the user-item graph structure by propagating embeddings on it. We argue that the existing methods cannot do fast adaption in the cold-start collaborative filtering setting, and they can be improved by our proposed method.

B. Cold-start Recommendation

Cold-start is a common problem in recommender systems when there is insufficient information in the recommendation process to make reliable recommendations for a user. It can be classified into a complete cold-start problem, where no interactions are available for new users, and incomplete cold-start problem, where there only exist few interactions for new users. In this work, we mainly deal with the incomplete cold-start case. Traditional collaborative filtering methods cannot solve this task without re-training the models. The standard way is to rely on content information to model new users' preferences and new items' characteristics. This has been widely used in many works [22], [23], [32]–[38]. The DropoutNet [22] trains deep neural network models to generalize to missing input by applying dropout, which can be viewed as a successful training method for pre-training the base models. MetaRec [36] presents a meta-learning strategy that learns the user's specific item representations and adapts biases of a neural network to address the cold-start issue on items. MeLU [23] proposes to use the state-of-the-art meta-learning to solve the user cold-start problem. Pan et al. [38] propose Meta-Embedding, a meta-learning based approach that learns to generate desirable initial embeddings for new ad IDs. Different from them, our MetaCF seeks to go beyond incorporating auxiliary features and utilizes structural information among users and items to learn the embeddings of new nodes. Therefore, MetaCF can solve the cold-start problem when the content information is unavailable, which is infeasible for the above models. Moreover, in [38] they need to pre-train the base-model, which is laborious. In contrast, our framework MetaCF can learn from scratch and do not need additional steps.

C. Meta-Learning

Meta-learning, also known as learning to learn, intends to design models that can learn new skills or adapt to new environments rapidly with a few training examples [39]. Previous work on the application of meta-learning in recommender systems includes [40]–[42]. Besides the works about cold-start recommendation we discussed before, there have been work [42] that utilized meta-learning to select recommendation algorithms and a recent work [40] proposes λ_{opt} to optimize regularization hyper-parameters based on the validation data.

VI. Conclusion and Future Work

In this paper, we proposed MetaCF which leverages meta-learning to address the cold-start problem in recommendation tasks. MetaCF can adapt to new users in an accurate and expeditious manner with only a few interactions. Extensive experiments on three real-world datasets demonstrate the effectiveness of MetaCF, which outperforms a set of state-of-the-art baselines. In future work, we would like to extend our work in the following three directions. First, we would like to migrate our model to solve the cold-start problems from the perspective of items. Second, we will extend our work to incorporate the relationships between users and items to construct a knowledge graph to further improve the effectiveness. Last, we would also like to explore the effectiveness of MetaCF in the scenario of sequential recommendation.

Acknowledgment

This work is partially supported by NSF III-1705169, NSF CAREER Award 1741634, NSF #1937599, DARPA HR00112090027, Okawa Foundation Grant, Amazon Research Award and the National Natural Science Foundation of China (61972372, U19A2079).

References

[1] S. Kabbur, X. Ning, and G. Karypis, "Fism: factored item similarity models for top-n recommender systems," in *SIGKDD*. ACM, 2013.
[2] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *SIGIR*. New York, NY, USA: ACM, 2019.
[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.
[4] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua, "Nais: Neural attentive item similarity model for recommendation," *IEEE TKDE*, vol. 30, no. 12, pp. 2354–2366, 2018.
[5] I. Fernández-Tobías, M. Braunhofer, M. Elahi, F. Ricci, and I. Cantador, "Alleviating the new user problem in collaborative filtering by exploiting personality information," *User Model User-adapt Interact*, 2016.
[6] L. Hu, S. Jian, L. Cao, Z. Gu, Q. Chen, and A. Amirbekyan, "Hers: Modeling influential contexts with heterogeneous relations for sparse and cold-start recommendation," in *AAAI*, vol. 33, 2019, pp. 3830–3837.
[7] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *CVPR*, June 2019.
[8] J. Gu, Y. Wang, Y. Chen, K. Cho, and V. O. Li, "Meta-learning for low-resource neural machine translation," *EMNLP*, 2018.
[9] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.
[10] X. Wang, X. He, F. Feng, L. Nie, and T.-S. Chua, "Tem: Tree-enhanced embedding model for explainable recommendation," in *WWW*, 2018.
[11] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009.
[12] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *NeurIPS Workshop on Meta-Learning*, 2017.
[13] A. Noguchi and T. Harada, "Image generation from small datasets via batch statistics adaptation," in *ICCV*, 2019, pp. 2750–2758.
[14] T. Shaikhina and N. A. Khovanova, "Handling limited datasets with neural networks in medical applications: A small-data approach," *Artif Intell Med*, 2017.
[15] M. Gao, L. Chen, X. He, and A. Zhou, "Bine: Bipartite network embedding," in *SIGIR*, 2018, pp. 715–724.
[16] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *WWW*, 2016.
[17] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *SIGKDD*. ACM, 2018, pp. 1059–1068.
[18] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *SIGIR*, 2017, pp. 335–344.
[19] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *SIGKDD*. ACM, 2018, pp. 1531–1540.
[20] Y. Tay, L. Anh Tuan, and S. C. Hui, "Latent relational metric learning via memory-based attention for collaborative ranking," in *WWW*, 2018.
[21] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," in *AAAI*, 2019.
[22] M. Volkovs, G. Yu, and T. Poutanen, "Dropoutnet: Addressing cold start in recommender systems," in *NeurIPS*, 2017, pp. 4957–4966.
[23] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation," in *SIGKDD*, 2019.
[24] Q. Wang, H. Yin, Z. Hu, D. Lian, H. Wang, and Z. Huang, "Neural memory streaming recommender networks with adversarial training," in *SIGKDD*, 2018, pp. 2467–2475.
[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2014.
[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.
[27] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.
[28] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.
[29] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," in *SIGKDD*, 2018.
[30] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*. ACM, 2018, pp. 974–983.
[31] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *RecSys*, New York, NY, USA, 2018.
[32] K. Mo, B. Liu, L. Xiao, Y. Li, and J. Jiang, "Image feature learning for cold start problem in display advertising," in *IJCAI*, 2015.
[33] S. Roy and S. C. Guntuku, "Latent factor representations for cold-start video recommendation," in *RecSys*, 2016, pp. 99–106.
[34] Y. Seroussi, F. Bohnert, and I. Zukerman, "Personalised rating prediction for new users using latent factor models," in *HyperText*, 2011.
[35] M. Zhang, J. Tang, X. Zhang, and X. Xue, "Addressing cold start in recommender systems: A semi-supervised co-training algorithm," in *SIGIR*, 2014, pp. 73–82.
[36] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *NeurIPS*, 2017, pp. 6904–6914.
[37] Z. Du, X. Wang, H. Yang, J. Zhou, and J. Tang, "Sequential scenario-specific meta learner for online recommendation," *SIGKDD*, 2019.
[38] F. Pan, S. Li, X. Ao, P. Tang, and Q. He, "Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings," in *SIGIR*, 2019, pp. 695–704.
[39] Z. Hu, T. Chen, K.-W. Chang, and Y. Sun, "Few-shot representation learning for out-of-vocabulary words," in *ACL*, Jul. 2019.
[40] Y. Chen, B. Chen, X. He, C. Gao, Y. Li, J.-G. Lou, and Y. Wang, "λopt: Learn to regularize recommender models in finer levels," in *SIGKDD*, 2019, pp. 978–986.
[41] H. Bharadhwaj, "Meta-learning for user cold-start recommendation," in *IJCNN*. IEEE, 2019, pp. 1–8.
[42] T. Cunha, C. Soares, and A. C. de Carvalho, "Meta learning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering," *Inf. Sci.*, 2018.