IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 December 2022, accepted 26 December 2022, date of publication 12 January 2023, date of current version 19 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3236391

RESEARCH ARTICLE

HyperRS: Hypernetwork-Based Recommender System for the User Cold-Start Problem

YUXUN LU[1,2,4], (Member, IEEE), KOSUKE NAKAMURA[3], AND RYUTARO ICHISE[2,4]

[1]The Graduate University for Advanced Studies, SOKENDAI, Shonan, Hayama, Kanagawa 240-0193, Japan
[2]Tokyo Institute of Technology, Meguro-ku, Tokyo 152-8552, Japan
[3]Information Technology Research and Development Center, Mitsubishi Electric Corporation, Kamakura, Kanagawa 247-8501, Japan
[4]National Institute of Informatics, Chiyoda-ku, Tokyo 101-8430, Japan

Corresponding author: Ryutaro Ichise (ichise@iee.e.titech.ac.jp)

ABSTRACT Meta-learning has been proven to be effective for the cold-start problem of recommender systems. Many meta-learning recommender systems that are designed for the user cold-start problem are gradient-based. They use a global parameter learned from existing users to initialize the recommender system parameter for new users that provides a personalized recommendation with limited user-item interactions. Many systems require users' demographic information to learn the global parameter. This requirement raises privacy concerns, and demographic information is not always available. In addition, some of the gradient-based systems need dozens or even hundreds of user-item interactions from existing users to learn the global parameter. This requirement is difficult to satisfy in specific scenarios in which active users and their item interactions are rare. Moreover, gradient-based meta-learning systems rarely capture user preferences over different item attributes, as updating the corresponding weights requires many optimization steps. We proposed HyperRS, a **Hyper**network-based **R**ecommender **S**ystem, for the user cold-start problem. Our system does not rely on demographic information to provide personalized recommendations. In our system, a hypernetwork generates all weights in the underlying recommender system. The hypernetwork enables the weights to adapt quickly to capture user interest in both item attributes and the contents of the item attributes. The experimental results show that our method outperforms several state-of-the-art meta-learning recommender systems for the user cold-start problem.

INDEX TERMS Meta-Learning, recommender system, hypernetwork, cold-start problem.

I. INTRODUCTION

The recommender system has become indispensable for many online services. The recommender system attempts to capture patterns of user preference and provide users with items satisfying their interest. An important issue in recommender systems is to learn user preferences with limited data. Although recommender systems that utilize deep learning [1], [2] or collaborative filtering [3], [4] have achieved great success, they cannot handle users that rarely interact with items [5] because they require sufficient user behaviors to model the user's interest. The extreme case is known as the user cold-start problem in which no user-item interaction is available.

The associate editor coordinating the review of this manuscript and approving it for publication was Pasquale De Meo.

Recent studies [6], [7], [8], [9], [10], [11], [12], [13], [14], [15] on recommender systems use meta-learning to alleviate the user cold-start problem. Most methods utilize the gradient-based strategy introduced in Model Agnostic Meta-Learning (MAML) [16]. There are two categories of parameters under the framework of MAML: (1) local parameter, which is the user-specific parameter in the recommendation model (a neural network) to provide attractive items for users; and (2) global parameter, which is used as the initial value of the local parameter. The global parameter is learned from existing users. The local parameter and the global parameter are optimized reciprocally to provide a new user with a recommender system with an initial parameter that can quickly adapt to the user's interest. The MAML-based recommender systems can yield acceptable recommendations for new users because the global parameter has been optimized

by item interactions from existing users, and these systems can produce personalized recommendations because the local parameter is user specific and will be updated by the user's behaviors after initialization.

There are several issues in meta-learning recommender systems that use the MAML framework. First, local parameters in the recommendation model for different users are initialized by the same global parameter. This homogeneity results in difficulty in capturing the interest of users whose preference is poorly represented by the initialization [13]. Research [6], [8], [13] uses the demographic information of users to adapt global parameters for users with different profiles. However, the user profile is not always available, and collecting the user profile raises privacy concerns. Moreover, the effectiveness of the user profile depends on its diversity and representativeness [17], [18]. A user profile that contains insufficient aspects is barely useful, as in this case, users with the same profile may have different preferences [13].

Second, it is difficult for recommender systems with the MAML framework to capture user interest over item attributes *per se*. Not all item attributes are equally important to users. For example, in the `Movie` domain, some users value the `Genre` property more than other properties. The contents of the `Genre` property (i.e., `Sci-fi`, `Comedy`, `Horror`, `Romantic`, ...) are decisive for these users. Other properties (`Casters`, `Directors`, ...) are less important. In this case, the embedding of contents in `Genre` should have more important weights than embeddings of contents in other properties. The MAML-based recommender systems need many records to update the corresponding local parameter to capture such a preference over item attributes after initialization. Because of the long tail phenomenon of users (i.e., most users only have a small number of interactions [5]), most users do not have enough interactions for the recommender system to update the local parameter.

Third, some MAML-based methods [7], [14], [15], [19] use complicated neural networks in their systems. These models require users with dozens or hundreds of records to optimize the global and local parameters. While these methods are effective in domains with plentiful data (for example, online shopping for leisure goods), they cannot fit scenarios where all users have scarce data (e.g., ''second-hand car'' and ''home decoration''). In addition, because of the long tail phenomenon of users, the local parameter cannot be well optimized if the underlying recommendation model is heavily parameterized [12].

We proposed a hypernetwork [20], [21]-based meta-learning recommender system.[1] Different from MAML-based meta-learning recommender systems that use the global parameter to initialize weights for users in the recommender system and optimize the weights by gradients from user-item interactions after initialization, the weights for users are generated by a hypernetwork in our model. In addition to the user embedding that will be fed into the neural network that predicts the recommendation score, we assign every user an embedding as the input to the hypernetwork to generate the user-specific weights in the neural network, which takes the contents of item attributes and the user embedding to predict the item score to the user. Our contributions are summarized as follows:

1) Our system can capture user interests with less data than the MAML-based recommender systems. The hypernetwork generates weights in the neural network recommending items to the user. Our system only needs to optimize the user embedding for the hypernetwork to rapidly adapt parameters for capturing user interest. Our system requires fewer data than MAML-based recommender systems that need to optimize all weights after initialization. Our system can adapt new item features (i.e., item attribute contents) with the help of existing item features.
2) Our system can reflect user interests in item attributes. In our method, a user embedding for the hypernetwork captures the user preference over item attributes, and the preference over the contents of the item attributes is captured by another user embedding for the neural network that recommends items. The corresponding transforming elements in the underlying network reflect user interests in attribute attributes. The transformed item attribute content embeddings reflect user interests in the item attribute contents.
3) Our system has better performance in the scenario where the numbers of user-item interactions are rare. MAML-based recommender systems need many existing users to learn the global parameter that initializes the recommender system for new users. Experiments show that our model has better performance when the existing users are rare.

II. RELATED WORK

The essential criterion for handling the cold-start problem in recommender systems is whether the model can provide satisfying recommendations for users by only a few item interactions. The cold-start problem exists for both users and items. The item cold-start problem occurs when new items are added, and no user has interacted with them. Similarly, the user cold-start problem occurs when new users register in the system because no item interaction is available. In the following sections, the first subsection introduces methods that attempt to solve the user cold-start problem. The second subsection contains a brief introduction to the hypernetwork and methods that generate parameters in the recommender system to alleviate the cold-start problem.

A. META-LEARNING METHODS FOR THE USER COLD-START PROBLEM

Many methods [6], [7], [8], [10], [13], [22] attempt to tackle the user cold-start problem by meta-learning. They adopt the model agnostic meta-learning framework [16]. A representative work is MeLU [10]. MeLU uses a neural network f_θ that

[1]The code is on our GitHub https://github.com/ichise-laboratory/hyperrs

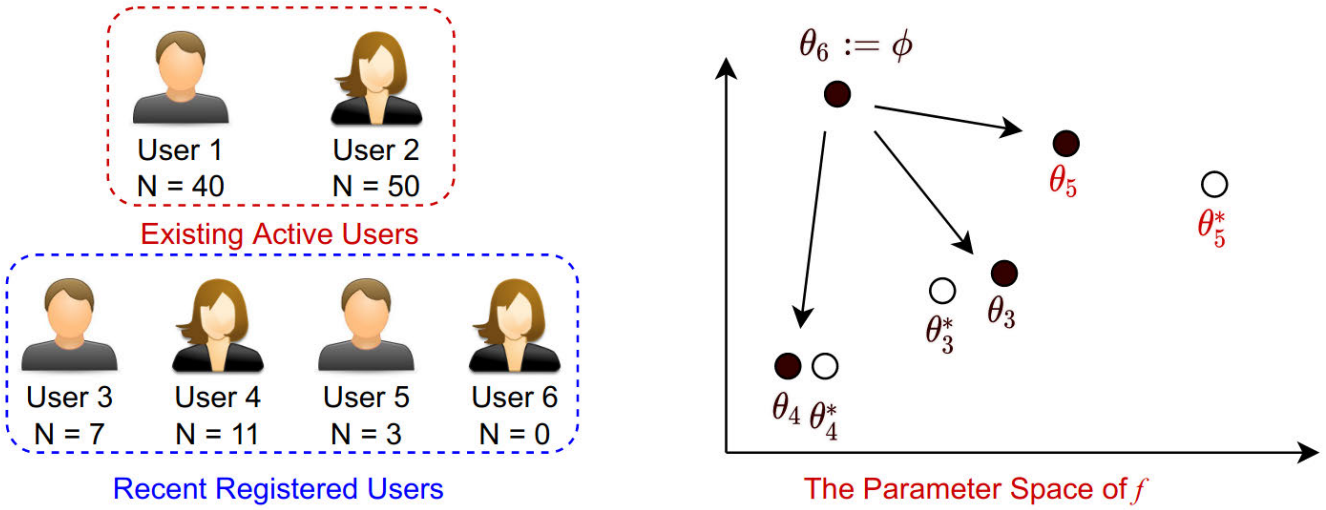


FIGURE 1. The MAML framework in the recommender system for the user cold-start problem. *N* is the number of item interactions for the user. θ_k^* is the optimal parameter for user *k*. θ_k is the parameter for user *k* after optimization by *N* item interactions.

takes user embeddings and item embeddings to predict the score of an item to a user. The user embeddings in MeLU are from users' side information (Age, Job, ...). The parameter θ in f_θ is user specific. It uses a global parameter ϕ learned from existing users by f_θ. For a recently registered user k, the initial parameter θ_k in the neural network f_{θ_k} is set to ϕ. θ_k is updated by user interactions from User k after the initialization. Because the parameter θ in the neural network f_θ is user specific, MeLU can provide personalized recommendation results for users, and the parameter θ_k for User k can quickly adapt to capture User k's interest because it is initialized by the global parameter ϕ, which has been optimized over existing users.

A critical problem in MeLU is that the number of item interactions may not be sufficient to optimize θ_k for User k. Fig. 1 shows an example of this problem (by User 5). User 3 and User 4 have more item interactions (7 and 11, respectively) than User 5 (3 item interactions). As a result, the user-specific parameters θ_3 and θ_4 can be optimized close to the optimal parameters (θ_3^*, θ_4^*), but the parameter θ_5 for User 5 still needs more optimization steps to reach the optimal parameter θ_5^* due to the deficient item interactions.

Many research attempts have been made to accelerate the local parameter optimization for news users in MeLU. MetaDNN [15] changes the neural network that predicts scores in MeLU to a deeper, wider version. It modifies the training procedure for ϕ to utilize the power of DNN to capture user interests with fewer interaction data. MAMO [6] employed a memory augmented neural network [23] to provide multiple global parameter ϕ's for users with different profiles and use an addressing algorithm to operate the memory so that the initialized θ in f_θ is closer to the optimal one. MetaHIN [8] uses meta-paths in a heterogeneous information network to extract user preference from directly interacting items with the information from the related items.

MeLU and MAMO need user profiles to construct user embeddings. In practice, this requirement raises privacy concerns. In addition, the user profile must contain diverse aspects of the demographic information for these methods. Otherwise, the performance deteriorates [13]. Some sequential/session-based recommender systems are proposed to alleviate the user cold-start problem without user profiles. s^2Meta [24] uses only user embeddings from user IDs. It induces an update gate and a stop gate in updating θ in f_θ and the global parameter ϕ. MetaTL [12] takes item embeddings as transitional vectors to capture the variance of user interest. A user's interest is described by all transitional vectors formed from items in MetaTL.

Other works use different approaches in meta-learning to handle the cold-start problem. For example, Zhu et al. proposed a method that uses the embeddings of existing items/users to optimize the embeddings of new items/users by shifting and scaling [25]. ProtoCF [26] integrates a prototypical network [27] to learn prototypes of items to solve the item cold-start problem.

B. PARAMETER GENERATION FOR THE COLD-START PROBLEM AND HYPERNETWORK

Recent studies [7], [28] suggest that using dynamic parameters in the neural network can improve the recommendation in the cold-start problem. CMML [7] uses weight modulation in the last layer of the neural network to decrease the impact of insignificant features from the previous layer. Raziperchikolaei et al. used dynamic parameters generated from items to handle the item cold-start problem [28]. In these works, only a portion of the parameters in the neural network that predicts item scores is generated. Hypernetwork [20] aims to generate *all* weights θ in a neural network f_θ by another neural network g. It can be integrated into a similar MAML training procedure as a meta-learning approach [29].

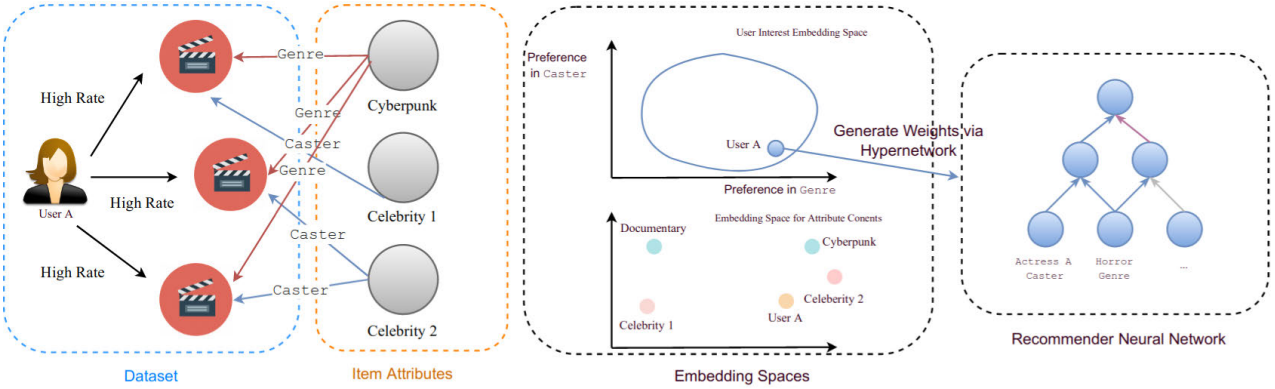


FIGURE 2. The example of HyperRS capturing the interest of User A on item attributes ("Genre" and "Caster") and contents in item attributes ("cyberpunk" in "Genre", "celebrity 1", "celebrity 2" in "Caster"). As shown in the embedding space part, user and item attribute content embeddings are in the same feature space to capture the user's interest in attribute contents. The interest in item attributes is captured by the hypernetwork and represented as the generated weights in the recommender network.

Shamsian et al. showed that the hypernetwork can be used in a similar role to the global parameter ϕ in the MAML framework to provide a personalized neural network for different devices in federated learning [21]. Lamb et al. showed that the hypernetwork can adapt to features rapidly rather than reusing them [30]. We adopt the idea of using a hypernetwork to provide a personalized neural network in meta-learning: every user is assigned a neural network with user-specific parameters. The hypernetwork provides user-specific parameters in the underlying neural network that recommend items to the user. In our method, the recommender system is in the form of a neural network f_θ. In contrast to the MAML-based approaches, which only use dynamic weights in a small part of the recommender network, our method generates *all parameters* in the recommender network by a hypernetwork g_ϕ. Details of our method are introduced in the next section.

III. PROPOSED METHOD

Our system contains two components: (1) a neural network (the recommender network) that takes user embeddings and item embeddings as input and predicts scores of items to the user; and (2) another neural network (the hypernetwork) that takes *user interest embeddings* and generates all parameters (weights) in the recommender network. The weights of transforming embeddings of the contents in the item attributes represent the users' interest in the item attributes. The recommender network takes the user embedding and embeddings of contents in item attributes to reflect the user interest over these contents.

An example in Fig. 2 explains the intuition in our method. User A in the example has interacted with three items (movies). Each movie is described by two attributes `Caster` and `Genre`. These attributes can be provided by the online platform or a knowledge graph. Because User A rates all items high scores if the items have content "cyberpunk" in the attribute `Genre` and 2 of 3 high score items are acted on by "celebrity 2" in the attribute `Caster`, our model will treat `Genre` and `Caster` as two essential item attributes to User A. The user interest embedding of User A will be optimized to make the hypernetwork generate larger weights in the recommender network for attribute content embeddings of `Caster` and `Genre`, as shown in the right part of Fig. 2. The circle in the user interest embedding space (the top-middle part in Fig. 2) is a subspace spanned by the *user interest basis*. The user interest embedding is a mixture of *user interest basis*. Details about embeddings in our method are introduced in the following subsections.

A. EMBEDDINGS IN HyperRS

We treat the user-item interactions as $\mathcal{D} = \{(u, i, s)|u \in \mathcal{U}, i \in \mathcal{I}, s \in \mathcal{S}\}$. $\mathcal{U}$ and $\mathcal{I}$ are the set of users and items, respectively. $\mathcal{S}$ is the set of scores. In most cases, $\mathcal{S}$ is a finite set that contains valid score values. If the score is in the range of a nonnegative real number, $\mathcal{S} = \mathbb{R}^+$.

Every user u has two embeddings: a user embedding $\mathbf{u}$ used in the recommender network and a user interest embedding $\hat{\mathbf{u}}$ used in the hypernetwork. $\mathbf{u}$ is formed by embedding-lookup of the user's id. $\hat{\mathbf{u}}$ is formed by mixing a group of user interest bases. The construction of $\hat{\mathbf{u}}$ is introduced in Section III-C.

Every item $i \in \mathcal{I}$ is described by a group of attributes $p(i) = \left(p_1^{(i)}, p_2^{(i)}, \ldots, p_k^{(i)}\right)$. k is the number of attributes. The contents of attributes are encoded as one-hot or multi-hot vectors $\left(\mathbf{p}_1^{(i)}, \mathbf{p}_2^{(i)}, \ldots, \mathbf{p}_k^{(i)}\right)$. The item attribute content embedding $\mathbf{e}_x^{(i)}$ is formed as

$$\mathbf{e}_x^{(i)} = \mathbf{M}_x \mathbf{p}_x^{(i)}; \qquad x = 1, 2, \ldots, k; \tag{1}$$

$\mathbf{M}_x$ contains embeddings for all contents in attribute x as column vectors.

B. RECOMMENDER NETWORK

The left part in Fig. 3 shows the structure of the recommender network. The recommender network predicts item scores for a user. The hypernetwork generates all parameters in the recommender network.

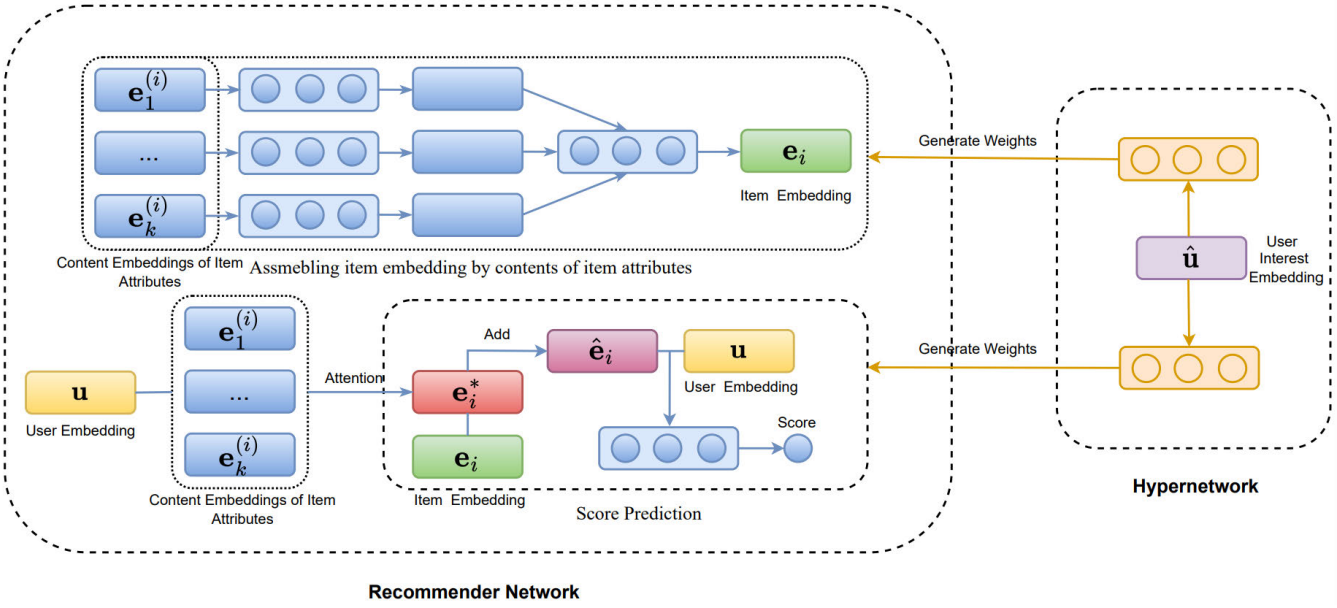


FIGURE 3. The structure of our network. The recommender network (left) transforms the item profile to item embedding and predicts the user's interest in the item with the user embedding and the item embedding. Hypernetwork (right) generates weights in the recommender network by user's interest embedding.

Item attribute content embeddings $\mathbf{e}_x^{(i)}$ are transformed by a hidden layer. The transformed embeddings are merged to form the item embedding $\mathbf{e}_i$:

$$\hat{\mathbf{e}}_x^{(i)} = \phi(\mathbf{W}_x \mathbf{e}_x^{(i)} + \mathbf{b}_x); \qquad x = 1, 2, \ldots, k. \tag{2}$$

$$\mathbf{e}_i = \phi\big(\mathbf{W}_t[\hat{\mathbf{e}}_1^{(i)} \oplus \ldots \oplus \hat{\mathbf{e}}_k^{(i)}] + \mathbf{b}_t\big). \tag{3}$$

$\oplus$ is the vector concatenation.$\mathbf{W}_x$, $\mathbf{W}_t$, $\mathbf{b}_t$ are weights, and $\mathbf{b}_x$, $\mathbf{b}_t$ are biases. ϕ is the activation function.

As mentioned above, the task of user embedding in the recommender network is to capture user interest in the contents of item attributes. This task is represented by a residual block [31] with attention in the recommender network:

$$\mathbf{e}_i^* = \text{attention}(\mathbf{u}, \mathbf{P}). \tag{4}$$

$$\hat{\mathbf{e}}_i = \mathbf{e}_i^* + \mathbf{e}_i. \tag{5}$$

$\mathbf{P}$ contains $\mathbf{e}_x^{(i)}$ as column vectors. $\mathbf{u}$ is the user embedding. The score is predicted by a 4-layer neural network f_θ:

$$\hat{s}^{(u,i)} = f_\theta(\hat{\mathbf{e}}_i \oplus \mathbf{u}). \tag{6}$$

The parameter $\theta = \{\mathbf{W}_{\ell_1}, \mathbf{b}_{\ell_1}, \mathbf{W}_{\ell_2}, \mathbf{b}_{\ell_2}, \mathbf{W}_{\ell_3}, \mathbf{b}_{\ell_3}, \mathbf{W}_{\ell_4}\}$ in f_θ contains weight $\mathbf{W}_{\ell_j}$ and bias $\mathbf{b}_{\ell_j}$ used in hidden layer ℓ_j. The last layer does not use the bias.

The loss function of our model is

$$\mathcal{L}(s^{(u,i)}, \hat{s}^{(u,i)}) = (s^{(u,i)} - \hat{s}^{(u,i)})^2. \tag{7}$$

$s^{(u,i)}$ is the score in the dataset, and $\hat{s}^{(u,i)}$ is the item i's score predicted by the recommender network for User u.

The parameter set $\Theta = \{\mathbf{W}_x, \mathbf{b}_x, \mathbf{W}_t, \mathbf{b}_t | x = 1, 2, \ldots, k\} \cup \theta$ includes all parameters in the recommender network.

C. HYPERNETWORK

The task of the hypernetwork is to generate the parameter in Θ. The input of the hypernetwork is the user interest embedding $\hat{\mathbf{u}}$:

$$\hat{\mathbf{u}} = \mathbf{M}_u \mathbf{u}^*. \tag{8}$$

$\mathbf{M}_u \in \mathbb{R}^{p \times q}$ and $\mathbf{u}^* \in \mathbb{R}^q$. $q < p$. The column vectors $\mathbf{m}_u$ in $\mathbf{M}_u$ are the user interest basis mentioned at the beginning of Section III-A. $\mathbf{u}^*$ is formed by embedding-lookup of the user's id.

Equation (8) inherits the motivation from latent embedding optimization (LEO) [32]. The intuition of (8) is to learn the user interest basis in $\mathbf{M}_u$ from existing users to enable fast optimization of the user interest embedding $\hat{\mathbf{u}}$ for new users by only changing elements in $\mathbf{u}^*$ during backpropagation. $\hat{\mathbf{u}}$ is a mixed component of all user interest bases in $\mathbf{M}_u$. The weights of the user interest basis are stored in $\mathbf{u}^*$ for building $\hat{\mathbf{u}}$. Because all parameters in the recommender network are generated by the hypernetwork that takes $\hat{\mathbf{u}}$ as the input, *the fast adaptation of* $\hat{\mathbf{u}}$ *by only changing* $\mathbf{u}^*$ *enables parameters in the recommender network to be generated appropriately at the same time.*

The components of the hypernetwork are one-layer neural networks:

$$\theta^* := \phi(\mathbf{W}_{\theta^*} \hat{\mathbf{u}} + \mathbf{b}_{\theta^*}) \qquad \theta^* \in \Theta. \tag{9}$$

The simple structure prevents the number of parameters in the hypernetwork from blowing since the number of parameters increases vastly with the number of layers: if θ^* is a $i \times j$ matrix (a weight in the recommender network), because the number of elements in $\hat{\mathbf{u}}$ is p, $\mathbf{W}_{\theta^*}$ will have $i \times j \times p$

parameters.[2] ϕ is the active function. The parameter set $\Phi = \{\mathbf{W}_{\theta^*}, \mathbf{b}_{\theta^*}, \mathbf{M}_u | \theta^* \in \Theta\}$ contains all parameters in the hypernetwork.

D. OPTIMIZATION

We follow the setting of optimization in meta-learning [16], [32]. Dataset $\mathcal{D}$ is divided into three subsets: $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{valid}}$ and $\mathcal{D}_{\text{test}}$. Users in these sets are not intersected. For a user u, we extract the support set $\mathcal{S}_u$ and the query set $\mathcal{Q}_u$. Items in $\mathcal{S}_u$ and $\mathcal{Q}_u$ are not intersected, but they may share the same contents in attributes.

For every user, the parameters for the recommender network are first initialized by the hypernetwork with the user interest embedding $\hat{\mathbf{u}}$. In training, the parameters in Θ are updated by gradient backpropagation with item interactions from $\mathcal{S}$. The recommender network with updated parameters is used on the query set $\mathcal{Q}$ to update the parameters in Φ.

We use negative sampling in optimization. Items that have not been interacted with users are negative samples with a score of 0. These negative examples are added into $\mathcal{S}$ and $\mathcal{Q}$ in training. The optimization procedure is described in Algorithm 1.

Algorithm 1 Optimization Procedure

Data: Support Set $\mathcal{S}$ and Query Set $\mathcal{Q}$. $\mathcal{S}, \mathcal{Q} \subset \mathcal{D}$.
Learning rates α, β.
Randomly initialize parameters $\mathbf{\Phi}$ in the hypernetwork;
for $i \leftarrow 1$ **to** N **do**
  for $u \in \mathcal{U}_{train}$ **do**
    Initialize parameters in $\mathbf{\Theta}$ by (8) and (9);
    for $(u, i, s) \in \mathcal{S}_u$ **do**
      Predict $\hat{s}^{(u,i)}$ by the recommender network;
      $\forall \boldsymbol{\gamma} \in \mathbf{\Theta} : \boldsymbol{\gamma} := \boldsymbol{\gamma} - \alpha \nabla_{\boldsymbol{\gamma}} \mathcal{L}(s, \hat{s}^{(u,i)})$;
      $\mathbf{u}^* := \mathbf{u}^* - \alpha \nabla_{\mathbf{u}^*} \mathcal{L}(s, \hat{s}^{(u,i)})$
    for $(u, i, s) \in \mathcal{Q}_u$ **do**
      Predict $\hat{s}^{(u,i)}$ by the recommender network;
      $\forall \boldsymbol{\gamma} \in \mathbf{\Phi} : \boldsymbol{\gamma} := \boldsymbol{\gamma} - \beta \nabla_{\boldsymbol{\gamma}} \mathcal{L}(s, \hat{s}^{(u,i)})$;
      $\mathbf{M}_x = \mathbf{M}_{\mathbf{x}} - \beta \nabla_{\boldsymbol{M}_x} \mathcal{L}(s, \hat{s}^{(u,i)})$;
      $\mathbf{u} = \mathbf{u} - \beta \nabla_{\mathbf{u}} \mathcal{L}(s, \hat{s}^{(u,i)})$;

IV. EXPERIMENT AND DISCUSSION

We conducted experiments in three datasets: Movielens-10m [33], BookCrossing [34] (named as "Book" hereafter), and an original dataset, Tokyo TV. It contains one month of watch logs of TV programs by residents in Tokyo 23 wards. We created the Tokyo TV dataset with data obtained from M Data Co., Ltd.

A. DATA PREPROCESSING AND EXPERIMENT SETTING

We extract attributes of movies in MovieLens-10m from WikiData so that both datasets contain the same attributes for items. Movies with incomplete features are discarded from the MovieLens-10m dataset. The score range for MovieLens-10m is from 0.5 to 5, with an interval of 0.5. The score range for Tokyo TV is from 1 to 3, with an interval of 1.

TABLE 1. Statistics of preprocessed Movielens-10m dataset and Tokyo TV dataset.

	Data	User	Item	Creator	Character	Genre
Movielens-10m	6,601,340	58,711	6,178	3,571	28,252	273
Tokyo TV	7,183	70	579	7	9,374	11
Book	580, 476	7,383	82,790	41,533	10,829	37,780

The data in Book dataset is a mixture of explicit and implicit interactions. The items that are interacted but not scored by users have a 0 score, whereas the explicit score ranges from 1 to 10 with an interval of 1. We normalize the scores in the Book dataset: all interacted items with a score of 1 because the score distribution is skewed: most explicitly scored items are with scores close to 10. The normalization eliminates the effect of skewed score distribution on all methods. Item attributes and the corresponding contents are extracted from the OpenLibrary data dump[3]: we take authors as `Character`, publishers as `Creator`, and subjects as `Genre`. The statistics of these three datasets are shown in Table 1.

We keep 20 records for every user in Movielens-10m and Tokyo TV datasets, and 10 for users in the Book dataset. The purpose of this setting is to simulate the scenario in which user-item interactions are rare with different scales of the number of users: MovieLens-10m and Book have a large number of users, whereas Tokyo TV (our original dataset) has few users. In addition, we use two experiment settings for Movielens-10m and Book datasets: cold users and warm items (C-W) and cold users with cold items (C-C.) In the cold users and warm items setting, all items in the validation and test datasets are in the training set, whereas in the cold users and cold items setting, only items that do not appear in the training set are used in the validation and test sets. The cold users with warm items experiments examine the capability of learning global parameters in the recommender network, whereas the cold users with cold items focus on examining the capability of fast adaption of item features. The experiment on Tokyo TV dataset only takes the cold users with warm item setting because the numbers of users and items do not support the cold item setting.

B. EVALUATION PROCEDURE AND PERFORMANCE MEASURE

We use binary normalized discounted cumulative gain (NDCG) and Recall@N to test model performance. Both measures are within the range [0, 1]. The binary NDCG@N

[2] Note that, as $\hat{\mathbf{u}}$ exists in the space spanned by column vectors in $\mathbf{M}_u$, the dimension of $\hat{\mathbf{u}}$ is q.

[3] cf. https://openlibrary.org/developers/dumps

TABLE 2. Experiment results on Movielens-10m dataset.

	NDCG@5		NDCG@10		NDCG@20		Recall@5		Recall@10		Recall@20	
	C-W	C-C	C-W	C-C	C-W	C-C	C-W	C-C	C-W	C-C	C-W	C-C
MeLU	0.2859	0.1540	0.3605	0.1615	0.4451	0.1811	0.2002	0.0946	0.3074	0.1051	0.4570	0.1404
s^2Meta	0.5884	0.4922	0.6782	0.5332	0.7056	0.5543	0.4366	0.3424	0.5588	0.3988	0.6082	0.4367
MetaTL	0.5657	0.4471	0.6507	0.4896	0.6618	0.4992	0.3844	0.2812	0.4923	0.3326	0.5109	0.3531
Mecos	0.4850	0.4633	0.6162	0.6377	0.6423	0.6493	0.3294	0.3224	0.5012	0.5529	0.5445	0.5723
HyperRS	**0.5893**	**0.4965**	**0.8834**	**0.7011**	**0.9026**	**0.7356**	**0.4533**	**0.3774**	**0.8699**	**0.6648**	**0.9037**	**0.7264**

TABLE 3. Experiment results on Book dataset.

	NDCG@5		NDCG@10		NDCG@20		Recall@5		Recall@10		Recall@20	
	C-W	C-C	C-W	C-C	C-W	C-C	C-W	C-C	C-W	C-C	C-W	C-C
MeLU	0.1524	0.0340	0.1861	0.0356	0.2086	0.0388	0.0592	0.0276	0.2014	0.0314	0.2542	0.0381
s^2Meta	0.3748	0.1388	0.3829	0.1398	0.3874	0.1443	0.2873	0.101	0.3021	0.1029	0.3127	0.1133
MetaTL	0.3691	0.1467	0.3892	0.1556	0.4039	0.1615	0.2479	0.0905	0.2796	0.1048	0.3092	0.1171
Mecos	**0.4196**	**0.3693**	**0.4285**	**0.3774**	**0.4386**	**0.3862**	0.2901	**0.2590**	0.3042	**0.2724**	0.3246	**0.2905**
HyperRS	0.2670	0.1727	0.3113	0.2156	0.3322	0.2189	**0.3077**	0.1752	**0.3852**	0.2514	**0.4345**	0.2590

TABLE 4. Experiment results on Tokyo TV dataset.

	NDCG@5	NDCG@10	NDCG@20	Hits@5	Hits@10	Hits@20
MeLU	0.3371	0.4135	0.4582	0.2384	0.3461	0.4307
s^2Meta	0.4435	0.4804	0.5268	0.3076	0.3615	0.4461
MetaTL	0.6474	0.788	0.8151	**0.5042**	0.7008	0.7521
Mecos	**0.6684**	0.9077	0.9260	0.4769	0.7923	0.8231
HyperRS	0.6285	**0.9158**	**0.9292**	0.4769	**0.8846**	**0.9077**

is computed as

$$\mathrm{DCG@N}(\mathbf{p}) = \sum_{i=1}^{N} \frac{2^{\mathbf{1}(p_i \in \mathbf{t})} - 1}{\log_2(i+1)}, \tag{10}$$

$$\mathrm{NDCG@N}(\mathbf{p}, \mathbf{t}) = \frac{\mathrm{DCG@N}(\mathbf{p})}{\mathrm{DCG@N}(\mathbf{t})}. \tag{11}$$

$\mathbf{p}$ is a set that contains N items with N highest predicted scores to a user. $\mathbf{t}$ is a set that contains items interacted by the user. $\mathbf{1}(s)$ is 1 if the statement s is true; otherwise, $\mathbf{1}(s) = 0$. The NDCG@N of a set $\mathcal{D}$ is averaged by the number of users in $\mathcal{D}$:

$$\mathrm{NDCG@N}(\mathcal{D}) = \frac{1}{|\mathcal{U}(\mathcal{D})|} \sum_{u \in \mathcal{U}(\mathcal{D})} \mathrm{NDCG@N}(\mathbf{p}(u), \mathbf{t}(u)). \tag{12}$$

$\mathcal{U}(\mathcal{D})$ denotes the set of users in $\mathcal{D}$. $\mathbf{p}(u)$ is the items with the top-N highest predicted score to User u. $\mathbf{t}(u)$ is the set of items interacted by u. We extract 60% users in the dataset for training, 20% for validation, and the last 20% for testing. In the validation and testing stages, the local parameter in MAML models and the parameter in our recommender network are reinitialized by the global parameter or the hypernet, and the support set data are used for updating the local parameter. The performance measures are computed with the data from the query set.

C. BASELINE MODEL AND HYPERPARAMETER SETTING

The activation functions in the recommender network and hypernetwork are ReLU and LeakyReLU, respectively. The user embeddings and item attribute content embeddings have 128 dimensions for Movielens-10m and Tokyo TV and 64 for Book. The user interest basis in the hypernetwork has 256 dimensions for Movielens-10m and Tokyo TV and 128 for Book, and the user interest embedding has 128 dimensions for Movielens-10m and Tokyo TV and 64 for Book.

We use the following four models as our baseline models:

1) MeLU [10] is a vanilla meta-learning recommender system based on the MAML [16] framework. It uses the user profile and item profile as user/item features. We replace the user profile with a user-specific embedding in their model.
2) s^2Meta [24] uses gates in updating local and global parameters to accelerate the optimization of the local parameter for new users. It utilizes the REINFORCE [35] algorithm to learn parameters for gates.
3) MetaTL [12] uses data in the support set to learn transition embeddings. These embeddings represent the user interest's transitions in items. MetaTL uses the previous k items in a sequence to predict the score of the $k+1$ item. MetaTL constitutes user interest transition embeddings by the k items in the user's support set and predicts the scores for items in the query set.

FIGURE 4. Results of all models with different sizes of support set and query set.

4) Mecos [36] learns the parameters in the recommender network that match the support set embedding to the query set embedding. It is akin to MetaTL: both methods use data in the support set to help the recommender network predict the user's interests in the query set items. It uses an LSTM cell [37] for the matching.

We train the MeLU and s^2Meta models as the instructions in the corresponding paper. Hyperparameters in the other models are optimized by grid search. We use Recall@5 as the criterion for choosing hyperparameters. The ranges are:

1) Learning rate α, β: $\{1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-4}, 5 \times 10^{-4}, 5 \times 10^{-5}, 6 \times 10^{-5}, 7 \times 10^{-5}\}$;
2) Embedding Size: {32, 50, 64, 100, 128, 256}.
3) Epoch num N: {50, 20, 5, 1}.

D. EXPERIMENT RESULT AND DISCUSSION

Table 2 to 4 show the results on the MovieLens-10m and Tokyo TV datasets, respectively. Results for Movielens-10m and Tokyo TV datasets have support set size $|\mathcal{S}| = 10$ and query set size $|\mathcal{Q}| = 10$. The Book dataset uses $|\mathcal{S}| = |\mathcal{Q}| = 5$.

Experiment results in Table 3 indicate that the hypernetwork can generate appropriate parameters in the recommender network under the cold users with warm items (C-W) setting on Book dataset. The best Recall@N measures for the C-W setting reflect that our method captures the user interests with adapted item attribute content embeddings. The lower NDCG@N (compared to Mecos) measures in the C-W setting indicate that, although the items interacted by users

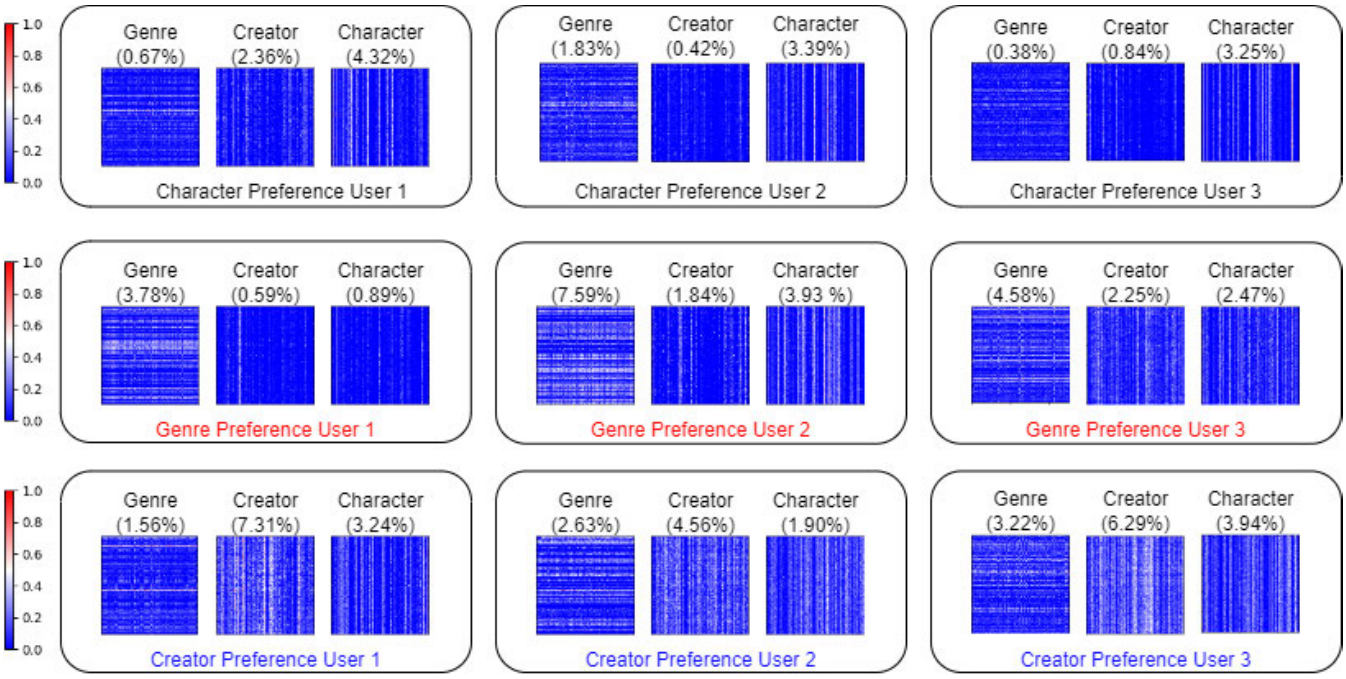


FIGURE 5. Heatmap of $\mathbf{W}_x$ in (2).

appear in the top N in prediction, the positions of these items are not very high. For example, 3 or 4 in Recall@5, or 7, 8, or 9 in Recall@10, cf. the equation for computing NDCG@N in (10) and (11). The C-C setting results in Table 3 show that HyperRS outperforms all other methods except for Mecos. The reason is that the number of item attribute contents in the Book dataset is much more significant than other datasets, cf. Table 1. Besides, the support and query set sizes are smaller than those for Movielens-10m and Tokyo TV datasets. As a result, the limited number of data (5 in the support and query sets) and the diverse item attribute contents make it difficult for our method to adapt the item attribute contents that did not appear in the training set. The Mecos learns the representations of the full support and query sets to match user interests with an LSTM cell to alleviate this problem.

Results in Table 2 and Table 4 indicate that under the setting of $|\mathcal{S}| = |\mathcal{Q}| = 10$, our method can better capture user interests with learned parameters in the hypernetwork. The generated parameters in the recommender network have a better performance compared to other models. For the cold users with warm items setting, HyperRS outperforms other models at least 1.67% on Recall@5 (0.4533 - 0.4366 for s^2Meta) and 0.09% (0.5893 - 0.5884 for s^2Meta) on NDCG@5 for Movielens-10m dataset, and it has the best performance on most measures for Tokyo TV dataset.

The C-C results on the Movielens-10m dataset and the Book dataset indicate that our method is more capable of capturing user interests with the parameters in the hypernetwork than adapting items with all new features. Because the item attribute contents in the Book dataset is much larger than those for the Movielens-10m dataset, the C-C setting contains more items with all new item attribute contents for all models in the Book dataset. Note that the cold items are chosen by the id of items rather than their attribute contents. In the Movielens-10m dataset, the `Creator` and `Genre` attributes are not as diverse as `Character`. The contents in those two attributes can help the adaption of new `Character` contents by the residual block in the recommender network, cf. (2) to (5).

Additionally, we further investigate the following research questions on the MovieLens-10m dataset.

RQ1 *How does HyperRS capture user interest over item attributes?* As we have mentioned at the beginning of Section III, the user interest in item attributes is reflected by the recommender network's weights. We normalize elements in $\mathbf{W}_x$ for converting item attribute content embeddings (cf. (2)) to range [0, 1]. Fig. 5 shows the heatmap for different $\mathbf{W}_x$. The numbers are the ratio of all weight elements larger or equal to 0.5. As Fig. 5 shows, different users have different categories of $\mathbf{W}_x$. For some users, the number of large values (>= 0.5 after normalization) in $\mathbf{W}_{\text{Creator}}$ is more than the other two weights $\mathbf{W}_{\text{Genre}}$ and $\mathbf{W}_{\text{Character}}$. Consequently, the information encoded in $\mathbf{p}_{\text{creator}}$ will be amplified in the item embedding $\mathbf{e}_i$ in (2). Therefore, users with this character can be categorized as `Creator` Preference User. The same rule is also applied for `Character` Preference User and `Genre` Preference User.

RQ2 *How does the performance vary with the support set's size?* In Section I, we stated that MAML-based methods may not be optimized if the global parameter is not appropriate or the optimization steps for the local parameter are insufficient. We test all models with different sizes of support set $|\mathcal{S}|$ and

query set $|\mathcal{Q}|$ on the MovieLens-10m dataset to investigate this issue. In addition to the result with $|\mathcal{S}| = 10, |\mathcal{Q}| = 10$ in Table 2, we also conduct experiments with $|\mathcal{S}| = 5$, $|\mathcal{Q}| = 15$ and $|\mathcal{S}| = 10, |\mathcal{Q}| = 5$ with the hyperparameters fine-tuned on the $|\mathcal{S}| = 10, |\mathcal{Q}| = 10$ setting. Fig. 4 shows the results with these settings. The name ''ssMeta'' refers to the s^2Meta model.

The results shown in Fig. 4 indicate that our method outperforms other models in all cases. In addition, the MAML-based methods require more data in the support set (cf. S=15 Q=5 in Fig. 4) to achieve similar performance compared to our method. Because the parameters in the recommender network are generated, therefore the parameter sharing in our method is indirect. It is different from what is in the MAML-based methods. Our method does not need to update the entire local parameter by gradients. Rather, our method updates the user interest embedding to generate these parameters. This property enables our method to fast adapt the parameters in the recommender network, as indicated from the study of Lamb et al. [30] and the results of C-W in Table 2 to 4.

V. CONCLUSION

We proposed HyperRS, a hypernetwork-based recommender system, to alleviate the user cold-start problem in recommender systems. HyperRS employs two neural networks, one f for predicting item score to users, and the other, g, generates parameters in f. The user interest over item attributes is reflected by the corresponding weights in f.

Experimental results on two datasets (one of which is collected by us) indicate that the hypernetwork can help parameters in the recommender network to quickly capture user interest. Compared to MAML-based approaches, our method has better recommendation performance. Additionally, our method can represent the user interest in item attributes with rare user-item interactions. Such interest is represented by the elements in weights from the recommender network.

In the future, we plan to investigate the relation between the structure of the hypernetwork and the performance of the recommender network. An essential problem in this investigation is to find a way of training the hypernetwork with exploding parameters - the number of parameters in the hypernetwork is vastly growing with the number of parameters in the recommender network. To fully utilize the power of the hypernetwork in the recommender system, we need to find a way of training these parameters.

The experimental results on different sizes of support set and query set indicate that our method needs fewer optimization steps compared to MAML-based methods. In addition, it does not require the demographic information from users to provide appropriate initial parameters in the recommender network for new users.

REFERENCES

[1] P. Zhao, K. Xiao, Y. Zhang, K. Bian, and W. Yan, ''AMEIR: Automatic behavior modeling, interaction exploration and MLP investigation in the recommender system,'' in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 2104–2110.

[2] W. Lee, K. Song, and I.-C. Moon, ''Augmented variational autoencoders for collaborative filtering with auxiliary information,'' in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 1139–1148.

[3] C. Chen, M. Zhang, Y. Zhang, W. Ma, Y. Liu, and S. Ma, ''Efficient heterogeneous collaborative filtering without negative sampling for recommendation,'' in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 19–26.

[4] J. Tang, F. Belletti, S. Jain, M. Chen, A. Beutel, C. Xu, and E. H. Chi, ''Towards neural mixture recommender for long range dependent user sequences,'' in *Proc. 28th World Wide Web Conf.*, 2019, pp. 1782–1793.

[5] J. Yin, C. Liu, W. Wang, J. Sun, and S. C. H. Hoi, ''Learning transferrable parameters for long-tailed sequential user behavior modeling,'' in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 359–367.

[6] M. Dong, F. Yuan, L. Yao, X. Xu, and L. Zhu, ''MAMO: Memory-augmented meta-optimization for cold-start recommendation,'' in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 688–697.

[7] X. Feng, C. Chen, D. Li, M. Zhao, J. Hao, and J. Wang, ''CMML: Contextual modulation meta learning for cold-start recommendation,'' in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2021, pp. 484–493.

[8] Y. Lu, Y. Fang, and C. Shi, ''Meta-learning on heterogeneous information networks for cold-start recommendation,'' in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 1563–1573.

[9] Y. Sun, K. Yin, H. Liu, S. Li, Y. Xu, and J. Guo, ''Meta-learned specific scenario interest network for user preference prediction,'' in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1970–1974.

[10] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, ''MeLU: Meta-learned user preference estimator for cold-start recommendation,'' in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1073–1082.

[11] H. Sun, J. Xu, K. Zheng, P. Zhao, P. Chao, and X. Zhou, ''MFNP: A meta-optimized model for few-shot next POI recommendation,'' in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 3017–3023.

[12] J. Wang, K. Ding, and J. Caverlee, ''Sequential recommendation for cold-start users with meta transitional learning,'' in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1783–1787.

[13] R. Yu, Y. Gong, X. He, B. An, Y. Zhu, Q. Liu, and W. Ou, ''Personalized adaptive meta learning for cold-start user preference prediction,'' in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 10772–10780.

[14] Y. Chen, X. Wang, M. Fan, J. Huang, S. Yang, and W. Zhu, ''Curriculum meta-learning for next POI recommendation,'' in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 2692–2702.

[15] H. Bharadhwaj, ''Meta-learning for user cold-start recommendation,'' in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[16] C. Finn, P. Abbeel, and S. Levine, ''Model-agnostic meta-learning for fast adaptation of deep networks,'' in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[17] S. Ahmadian, M. Afsharchi, and M. Meghdadi, ''A novel approach based on multi-view reliability measures to alleviate data sparsity in recommender systems,'' *Multimedia Tools Appl.*, vol. 78, no. 13, pp. 17763–17798, Jul. 2019.

[18] F. Tahmasebi, M. Meghdadi, S. Ahmadian, and K. Valiallahi, ''A hybrid recommendation system based on profile expansion technique to alleviate cold start problem,'' *Multimedia Tools Appl.*, vol. 80, no. 2, pp. 2339–2354, Jan. 2021.

[19] L. Wang, B. Jin, Z. Huang, H. Zhao, D. Lian, Q. Liu, and E. Chen, ''Preference-adaptive meta-learning for cold-start recommendation,'' in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1607–1614.

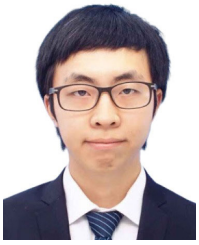
[20] D. Ha, A. Dai, and Q. V. Le, ''Hypernetworks,'' in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–29.

[21] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, ''Personalized federated learning using hypernetworks,'' in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 9489–9502.

[22] Y. Bi, L. Song, M. Yao, Z. Wu, J. Wang, and J. Xiao, ''DCDIR: A deep cross-domain recommendation system for cold start users in insurance domain,'' in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1661–1664.

[23] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, ''Meta-learning with memory-augmented neural networks,'' in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1842–1850.

[24] Z. Du, X. Wang, H. Yang, J. Zhou, and J. Tang, "Sequential scenario-specific meta learner for online recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2895–2904.
[25] Y. Zhu, R. Xie, F. Zhuang, K. Ge, Y. Sun, X. Zhang, L. Lin, and J. Cao, "Learning to warm up cold item embeddings for cold-start recommendation with meta scaling and shifting networks," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1167–1176.
[26] A. Sankar, J. Wang, A. Krishnan, and H. Sundaram, "ProtoCF: Prototypical collaborative filtering for few-shot recommendation," in *Proc. 15th ACM Conf. Recommender Syst.*, Sep. 2021, pp. 166–175.
[27] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4080–4090.
[28] R. Raziperchikolaei, G. Liang, and Y.-J. Chung, "Shared neural item representations for completely cold start problem," in *Proc. 15th ACM Conf. Recommender Syst.*, Sep. 2021, pp. 422–431.
[29] D. Zhao, J. V. Oswald, S. Kobayashi, J. A. Sacramento, and B. F. Grewe, "Meta-learning via hypernetworks," in *Proc. 4th Workshop Meta-Learn. NeurIPS*, 2020, pp. 300–311.
[30] A. Lamb, E. Saveliev, Y. Li, S. Tschiatschek, C. Longden, S. Woodhead, J. M. Hernández-Lobato, R. E. Turner, P. Cameron, and C. Zhang, "Contextual hypernetworks for novel feature adaptation," in *Proc. 4th Workshop Meta-Learn. NeurIPS*, 2020, pp. 30–41.
[31] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image steganalysis," *Multimedia Tools Appl.*, vol. 77, no. 9, pp. 10437–10453, May 2018.
[32] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *Proc. 5th Int. Conf. Learn. Represent.*, 2019, pp. 1–17.
[33] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015.
[34] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proc. 14th Int. Conf. World Wide Web*, 2005, pp. 22–32.
[35] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Lang.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.
[36] Y. Zheng, S. Liu, Z. Li, and S. Wu, "Cold-start sequential recommendation via meta learner," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 4706–4713.
[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.



YUXUN LU (Member, IEEE) received the B.E. degree in software engineering from Beijing Jiaotong University, China, and the M.E. degree in computer science from the Nara Institute of Science and Technology, Japan. He is currently pursuing the Ph.D. degree with the National Institute of Informatics and The Graduate University for Advanced Studies, Japan.

He is also a Research Assistant with the Tokyo Institute of Technology. His research interests include machine learning, representation learning, and knowledge discovery.

KOSUKE NAKAMURA received the B.E. and M.E. degrees from the School of Engineering, Tohoku University, Sendai, Japan, in 2017 and 2019, respectively.

He is currently a Researcher with Mitsubishi Electric Corporation. His research interests include machine learning, knowledge graph, and natural language processing.

RYUTARO ICHISE received the Ph.D. degree in computer science from the Tokyo Institute of Technology, Tokyo, Japan, in 2000. In 2000, he joined the National Institute of Informatics, Japan, as a Faculty Member. From 2001 to 2002, he was a Visiting Scholar at Stanford University. He is currently a Professor with the Tokyo Institute of Technology. His research interests include machine learning, semantic web, and data mining.