ColdU: User Cold-start Recommendation with User-specific Modulation

Anonymous Authors

***Abstract*—Crafting personalized recommendations for users with minimal interaction histories, a prevalent challenge in user cold-start recommendation within recommendation systems (RSs), is characterized by its pervasive nature. This issue is particularly pronounced in modern over-parameterized RSs built on deep networks, heightening the risk of overfitting for cold-start users. The significance of addressing the user cold-start problem extends to user satisfaction, platform growth, and ongoing algorithmic evolution. Recent approaches have modeled this challenge as a few-shot learning task, intending to rapidly generalize to personalized recommendations with limited training samples. However, existing methods are hampered by a high risk of overfitting and the substantial computational cost associated with learning large deep models. In response, this paper introduces ColdU, an innovative approach that leverages the capabilities of a multi-layer perceptron (MLP) to effectively approximate complex functions. To achieve parameter efficiency in modulating sample embeddings, the same MLP is employed for each element of the embeddings, with distinct MLPs used for different layers of the predictor. This design maintains the flexibility of MLPs while reducing the size of learnable parameters, facilitating easy personalization of recommendation models for cold-start users. Extensive experiments conducted on benchmark datasets consistently validate ColdU as a state-of-the-art solution, underscoring its efficacy in providing personalized recommendations for users with limited interaction histories.**

***Index Terms*—user cold-start recommendation, few-shot learning, meta learning**

I. Introduction

Recommendation systems (RSs) [1] target at providing suggestions of items that are most pertinent to a particular user, such as movie recommendation [2] and book recommendation [3]. Nowadays, RSs are abundant online, offering enormous users convenient ways to shop regardless of location and time, and also providing intimate suggestions according to their preferences. However, user cold-start recommendation [4] remains a severe problem in RSs. On the one hand, the users in RSs follow the long tail effect [5], with some users having just a few interaction histories. On the other hand, new users continuously emerge, naturally having rated only a few items in RSs. This problem is even more challenging as modern RSs are mostly built with over-parameterized deep networks, requiring a substantial amount of training samples for good performance and risking overfitting for cold-start users [6]. The user cold-start recommendation problem is of paramount importance in recommendation systems due to its direct impact on user experience, platform growth, and algorithmic advancements. Successfully addressing the cold-start challenge contributes to enhanced user satisfaction, positively impacting user retention and engagement. As online platforms continue to grow, accommodating diverse user behaviors and preferences becomes essential, necessitating personalized recommendation strategies. Moreover, a robust solution to the cold-start problem not only influences revenue and customer satisfaction for businesses but also drives algorithmic evolution, fostering advancements in recommendation methodologies. The ongoing research and innovation spurred by the user cold-start problem contribute not only to addressing the challenges of new user scenarios but also to the continual improvement of recommendation algorithms for both new and existing users.

Recently, a number of approaches model user cold-start recommendation problem as a few-shot learning problem [7]. Few-shot learning aims to rapidly generalize to new tasks, specifically personalized recommendations for cold-start users, with limited training samples, i.e., sparse interaction histories. Several works [8]–[12] employ the classic gradient-based meta-learning strategy, Model-Agnostic Meta-Learning (MAML) [13]. MAML learns an effective initialized parameter from a set of tasks and adapts it to a new task by performing a few steps of gradient descent updates on a restricted number of labeled samples. This line of models has shown great promise in alleviating the user cold-start problem. However, the challenge lies in the expertise required to fine-tune the optimization procedure to prevent overfitting, and there are concerns regarding the potentially lengthy inference time associated with gradient-based meta-learning strategies.

Another approach in addressing the user cold-start recommendation problem involves the use of hypernetworks [14] to directly map user interaction histories to user-specific parameters, a concept explored in works such as [9], [15]–[17], [30]. These modulation-based methods typically consist of an embedding layer, modulator, and predictor. The modulator is responsible for generating user-specific parameters, which then modulate the predictor using a modulation function. For instance, MAMO [9] employs external memory to guide personalized parameter initialization, TaNP [15] learns to map user interaction history to generate user-specific parameters, CMML [16] utilizes the same set of user-specific parameters across different layers, and PNMTA [17] additionally leverages a pretrained encoder to capture generalized representation. These approaches typically adopt feature-wise linear modulation function (FiLM) [18] to modulate the representation through scaling and shifting based on conditioning information. In contrast, the state-of-the-art method ColdNAS [30] suggests searching for the right functions at the right positions to modulate. However, it is worth noting that these methods often require learning additional large networks to guide the learning of recommendation models, potentially incurring additional computational and storage expenses.

In this paper, we introduce ColdU, a novel approach designed to address the user cold-start recommendation problem by incorporating user-specific modulation. Our method begins by encoding the task context, followed by the generation of user-specific adaptive parameters, which are utilized to modulate the predictor in a layer-wise fashion. In terms of modulation, we harness the capabilities of a multi-layer perceptron (MLP) to effectively approximate complex functions. To achieve parameter efficiency in modulating sample embeddings, we employ the same MLP for each element of sample embedding, with distinct MLPs utilized for different layers of the predictor. This design preserves the flexibility of MLPs while reducing the size of learnable parameters, facilitating easy personalization of recommendation models for cold-start users. Our extensive experiments on benchmark datasets addressing the user cold-start problem consistently illustrate that ColdU outperforms existing methods, achieving state-of-the-art performance.

II. Preliminaries on User Cold-Start Recommendation

A. Related Works

Crafting personalized recommendations for cold-start users poses a significant challenge, given their limited interaction histories [4]. Historically, collaborative filtering (CF)-based approaches [23]–[25], which predict user-item interactions by capturing relationships in a low-dimensional space, have demonstrated leading performance in recommendation systems (RSs). Nevertheless, these CF-based methods rely solely on the user's historical data, rendering them ineffective in addressing the user cold-start problem. To mitigate this challenge, content-based methods leverage user/item features [4] or even incorporate user social relations [26] to enhance predictions for cold-start users. A recent deep learning model, DropoutNet [6], employs a neural network with a dropout mechanism applied to input samples for inferring missing data. However, extending these content-based methods to accommodate new users, often necessitates retraining the model, posing a potential limitation.

A recent trend in addressing the user cold-start problem involves framing it as a few-shot learning problem [7]. The resulting models are designed to swiftly generalize and provide recommendations for new users with limited interaction histories. Many of these approaches adhere to the classical gradient-based meta-learning strategy, such as Model-Agnostic Meta-Learning (MAML) [13]. In this strategy, an initial set of parameters is learned from training tasks, followed by local updates on provided interaction histories through gradient descent. Several existing works explore different directions to enhance performance. MeLU [8] selectively adapts model parameters during the local update stage, while MetaCS [27] opts to adapt all model parameters. MAMO [9] introduces external memory to guide model adaptation, and MetaHIN [10] employs heterogeneous information networks to leverage rich semantics between users and items. REG-PAML [11] suggests using a user-specific learning rate during local updates, and PAML [12] incorporates social relations to share information among similar users.

While gradient-based meta-learning approaches exhibit adaptability to training data, they are computationally inefficient during test-time and often require expert tuning of the optimization procedure to prevent overfitting. A more recent approach involves hypernetwork-based methods, where a network guides the learning of recommendation models. TaNP [15] learns to map item interaction records to modulate item-specific parameters, while ColdNAS [30] identifies the proper modulation function and position through neural architecture search. However, both methods necessitate learning relatively large networks, introducing additional learning burdens.

B. Problem Formulation

In user cold-start recommendation problem, we focus on user u_i who only has rated a few items. Following recent works [10], [15], [27], we model the user cold-start recommendation problem as a few-shot learning problem. The target is to learn a model from a set of training user cold-start tasks $\mathcal{T}^{\text{train}}$ and generalize to provide personalized recommendation for new tasks.

In this paper, scalars are represented by lowercase letters, vectors are denoted by lowercase boldface letters, and matrices are denoted by uppercase boldface letters. Each task T_i corresponds to a user u_i. It has a support set $\mathcal{S}_i = \{(v_j, y_{i,j})\}_{j=1}^{N_s}$ containing existing interaction histories which records user u_i gives rating $y_{i,j}$ to item v_j. It also has a query set $\mathcal{Q}_i = \{(v_j, y_{i,j})\}_{j=1}^{N_q}$ containing interactions to predict. N_s and N_q are the number of interactions in $\mathcal{S}_i$ and $\mathcal{Q}_i$. In user cold-start recommendation, N_s is small.

III. The Proposed ColdU

In this section, we introduce the proposed ColdU, whose architecture is plotted in Figure 1. We first present the details of key components of ColdU: embedding layers, and an user-specific predictor. Then, we describe the learning and inference procedure of ColdU.

A. Embedding Layer

Both users and items can associate with several content features. Following existing models [9], [15], [17], we first use embedding layer E with parameter $\boldsymbol{\Theta}_E$ to embed the categorical features from users and items into dense vectors, i.e., $(\boldsymbol{u}_i, \boldsymbol{v}_j) = E(u_i, v_j; \boldsymbol{\Theta}_E)$.

For each u_i, we get a content embedding for each categorical content feature, and concatenate them into the initial user embedding. Given B user contents, the user embedding of u_i is obtained as:

$$\boldsymbol{u_i} = [\ \boldsymbol{W}_U^1 \boldsymbol{c}_i^1 \mid \boldsymbol{W}_U^2 \boldsymbol{c}_i^2 \mid \cdots \mid \boldsymbol{W}_U^B \boldsymbol{c}_i^B\ ], \tag{1}$$

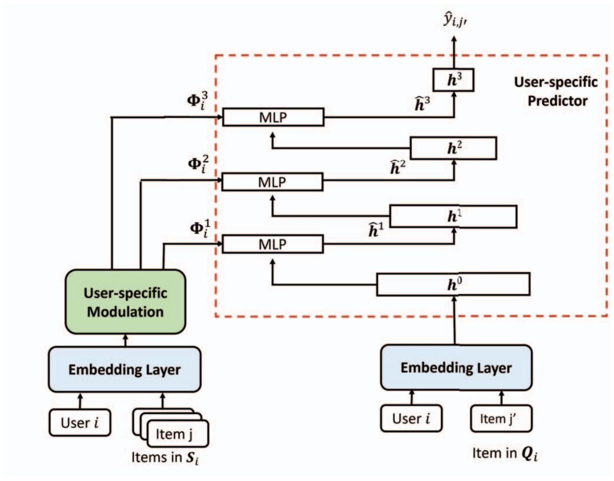


Fig. 1. An illustration of the proposed ColdU.

where $[\,\cdot \mid \cdot\,]$ is the concatenation operation, $\boldsymbol{c}_i^b$ is the one-hot vector of the bth categorical content of u_i , and $\boldsymbol{W}_E^b$ represents the embedding matrix of the corresponding content feature in the shared user feature space. Then, given B' item contents, the item embedding $\boldsymbol{v}_j$ for an item v_j of is similarly obtained using embedding layers with parameters $\{\boldsymbol{W}_V^b\}_{b=1}^{B'}$.

Collectively, $\boldsymbol{\Theta}_E = \{\boldsymbol{W}_U^1, \dots, \boldsymbol{W}_U^B, \boldsymbol{W}_V^1, \dots, \boldsymbol{W}_V^{B'}\}$ denotes the parameters of embedding layer of both users and items.

B. User-specific Predictor

Usually, a multi-layer perception (MLP) is used as P [15], [25], [28]. Hence, we design our user-specific predictor upon the MLP. Assume a L-layer MLP is used. Let $\boldsymbol{h}^l$ denote its output from the lth layer, and let $\boldsymbol{h}^0 = [\boldsymbol{u}_i \mid \boldsymbol{v}_j]$.

We first aim to map the user interaction histories of user u_i into a task context embedding $\mathbf{c}_i$. Specifically, We use a fully connected layer to get representation $\boldsymbol{r}_{i,j}$ of each interaction history $\boldsymbol{r}_{i,j}$:

$$\boldsymbol{r}_{i,j} = \text{ReLU}(\boldsymbol{W}_R[\ \boldsymbol{u}_i \mid \boldsymbol{v}_j \mid y_{i,j}\ ] + \boldsymbol{b}_R), \tag{2}$$

where $\boldsymbol{W}_R$ and $\boldsymbol{b}_R$ are learnable parameters. The task context embedding $\boldsymbol{c}_i$ of u_i is obtained by aggregating the interactions in $\mathcal{S}_i$ via mean pooling:

$$\boldsymbol{c}_i = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{r}_{i,j}. \tag{3}$$

To provide personalized recommendation for each u_i, we then generate user-specific adaptive parameters using $\mathbf{c}_i$. At the lth layer of predictor, we use a fully connected layer to map $\boldsymbol{c}_i$ to user-specific parameter $\boldsymbol{\phi}_i^l$ as

$$\boldsymbol{\phi}_i^l = \boldsymbol{W}_M^l \boldsymbol{c}_i + \boldsymbol{b}_M^l. \tag{4}$$

Subsequently, how to use the user-specific parameter $\boldsymbol{\phi}_i$ to change the prediction process can be crucial to the performance. As MLP can approximate a wide range of functions [29], we use it to modulate $\boldsymbol{h}^{l-1}$ in a parameter-efficient way. Specifically, the kth element of $\boldsymbol{h}^{l-1}$ is modulated by $\boldsymbol{\phi}$ as

$$\hat{\boldsymbol{h}}^l[k] = \text{ReLU}\left(\boldsymbol{w}_P^l(\boldsymbol{h}^{l-1}[k] \mid \boldsymbol{\phi}_i^l[k]) + b_P^l\right), \tag{5}$$

where $\boldsymbol{w}_P^l$ and b_P^l are parameters shared across elements $\boldsymbol{h}^{l-1}[1], \dots, \boldsymbol{h}^{l-1}[N_d]$ for $\boldsymbol{h}^{l-1} \in \mathbb{R}^{N_d}$. While different layers of predictor use different $(\boldsymbol{w}_P^l, b_P^l)$s.

Then, $\boldsymbol{h}^l$ is obtained from the modulated $\hat{\boldsymbol{h}}^l$ as

$$\boldsymbol{h}^l = \text{ReLU}(\boldsymbol{W}_Q^l \hat{\boldsymbol{h}}^l + \boldsymbol{b}_Q^l). \tag{6}$$

When $l = L$, $\hat{y}_{i,j} = \boldsymbol{h}^L$ is the final prediction.

For notation simplicity, let $\boldsymbol{\Theta}_P$ represents the parameters of the user-specific predictor, including $\boldsymbol{W}_R, \boldsymbol{b}_R$ in (2), $\{\boldsymbol{W}_M^l, \boldsymbol{b}_M^l\}_{l=1}^L$ in (4), $\{\boldsymbol{w}_P^l, \boldsymbol{b}_P^l\}_{l=1}^L$ in (5), and $\{\boldsymbol{W}_Q^l, \boldsymbol{b}_Q^l\}_{l=1}^L$ in (6).

C. Learning and Inference

For task $T_i \in \mathcal{T}^{\text{train}}$ associated with user u_i, we obtain the feature embeddings for u_i and v_js which are rated by u_i by embedding layers. Then, the user interaction histories recorded in $\mathcal{S}_i$ are mapped to user-specific parameters $\{\boldsymbol{\phi}_i^l\}_l = 1^L$, which are then used to modulate the prediction process. For each $(v_k, y_{i,j'}) \in \mathcal{Q}_i$, let $\boldsymbol{h}^0 = [\boldsymbol{u}_i \mid \boldsymbol{v}_{j'}]$. We obtain its sample representation by (5) and (6), and return $\hat{y}_{ij'}$ as the final prediction. The loss between the prediction $\hat{y}_{i,j}$ and true label $y_{i,j}$ is calculated by mean squared error (MSE):

$$\mathcal{L}_i = \frac{1}{N_q}\sum_{j'=1}^{N_q}(y_{i,j'} - \hat{y}_{i,j'})^2. \tag{7}$$

The model parameters $\boldsymbol{\Theta}_E, \boldsymbol{\Theta}_P$ of ColdU are optimized with respect to the following objective:

$$\mathcal{L}^{\text{train}} = \sum_{T_i \in \mathcal{T}^{\text{train}}} \mathcal{L}_i. \tag{8}$$

Algorithm 1 shows the complete training procedure.

During inference, consider a new task $T_{i'}$ corresponds to a new user $u_{i'}$, its support set $\mathcal{S}_{i'}$ and query set $\mathcal{Q}_{i'}$ are provided. we directly apply ColdU with the optimized $\boldsymbol{\Theta}_E^*, \boldsymbol{\Theta}_P^*$. Specifically, we take step 4-17 in Algorithm 1 to obtain prediction for each $(v_{j'}, y_{i',j'}) \in \mathcal{Q}_{i'}$.

IV. Experiments

We perform experiments on three benchmark datasets to evaluate the effectiveness of the proposed ColdU.

Experiments were conducted on a 24GB NVIDIA GeForce RTX 3090 GPU, with Python 3.7.0, CUDA version 11.6. Results are averaged over five runs.

TABLE I
SUMMARY OF BENCHMARK DATASETS USED IN THIS PAPER.

Dataset	# User (The Ratio of Cold-start Users)	# Item	# Rating	User Feature	Item Feature
MovieLens	6040 (52.3%)	3706	1000209	gender, age, occupation, Zip code	publication year, rate, genre, director and actor
BookCrossing	278858 (18.6%)	271379	1149780	age, location	year, author, publisher
Last.fm	1872 (15.3%)	3846	42346	user ID	item ID

Algorithm 1 Training procedure of ColdU.

```
Input: hyperparameter $L$;
1: randomly initialize all parameters in $\mathbf{\Theta}_E, \mathbf{\Theta}_P$;
2: while not converge do
3:   for every $T_i \in \mathcal{T}^{\text{train}}$ do
4:     for every $(v_j, y_{i,j}) \in \mathcal{S}_i$ do
5:       get representation $\mathbf{r}_{i,j}$ of $(v_j, y_{i,j})$ by (2);
6:     end for
7:     obtain task context embedding $\mathbf{c}_i$ of $\mathcal{S}_i$ by (3);
8:     for $l = 1, \dots, L$ do
9:       map $\mathbf{c}_i$ to user-specific parameter $\boldsymbol{\phi}_i^l$ by (4);
10:    end for
11:    for Every $(v_{j'}, y_{i,j'}) \in \mathcal{Q}_i$ do
12:      initialize sample embedding $\boldsymbol{h}^0 = [\boldsymbol{u}_i \mid \boldsymbol{v}_{j'}]$;
13:      for $l = 1, \dots, L$ do
14:        modulate embedding as $\hat{\boldsymbol{h}}^l$ by (5);
15:        update sample embedding as $\boldsymbol{h}^l$ by (6);
16:      end for
17:      return $\hat{y}_{i,j'}$;
18:    end for
19:    calculate loss $\mathcal{L}_i$ by (7).
20:  end for
21:  calculate the objective $\mathcal{L}^{\text{train}}$ by (8);
22:  optimize $\mathbf{\Theta}_E, \mathbf{\Theta}_P$ with respect to $\mathcal{L}^{\text{train}}$ by gradient
     descent;
23: end while
24: return optimized $\mathbf{\Theta}_E^*, \mathbf{\Theta}_P^*$.
```

A. Datasets

We use three benchmark datasets (Table I):

- **MovieLens**[1] [2]: a dataset containing 1 million movie ratings of users collected from MovieLens;
- **BookCrossing**[2] [3]: a dataset containing users' ratings on books in BookCrossing community;
- **Last.fm**[3]: a dataset containing user's listening count of artists from Last.fm online system. Negative samples for the query sets are generated following [15].

We split data following the recent state-of-the-art ColdNAS [30]. The ratio of $\mathcal{T}^{\text{train}} : \mathcal{T}^{\text{val}} : \mathcal{T}^{\text{test}}$ is set as $7:1:2$. The users associating with $\mathcal{T}^{\text{train}}, \mathcal{T}^{\text{val}}, \mathcal{T}^{\text{test}}$ are not overlapped. For MovieLens and Last.fm, users whose interaction history length lie in $[40, 200]$ are kept. Each support set is comprised of N=20 randomly selected user interactions, while the query set consists of the remaining interactions for the same user. In the case of BookCrossing, which exhibits a severe long-tail distribution of user-item interactions,users whose interaction history length falls within the range [50,1000) are put into the training set, denoted as $\mathcal{T}^{\text{train}}$. Subsequently, users with interaction history lengths in the range [2,50) are divided into 70%, 10%, and 20%, and assigned to $\mathcal{T}^{\text{train}}$, $\mathcal{T}^{\text{val}}$, and $\mathcal{T}^{\text{test}}$ respectively. For each user, half of their interaction history are randomly sampled to form the support set, while the remaining interactions constitute the query set.

B. Baselines

We compare **ColdU** with the following representative user cold-start methods. We run the public codes provided by the respective authors if they are available.

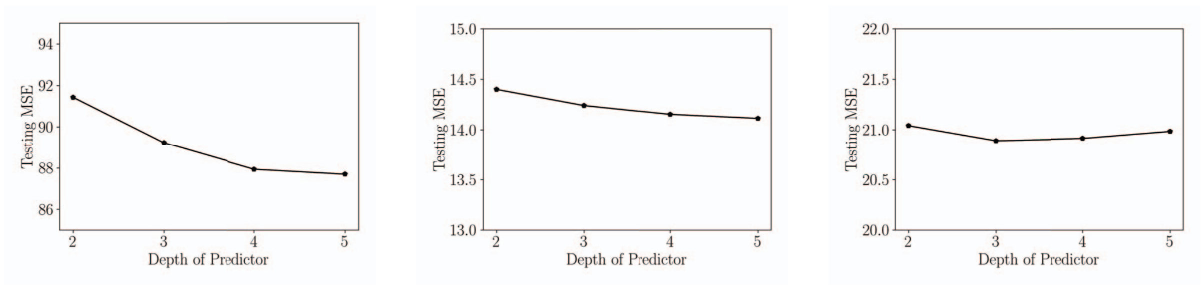
- **DropoutNet**[4] [6]: a traditional deep cold-start model which randomly dropouts preference informations during training.
- **MeLU**[5] [8]: a gradient-based meta learning method adapted from MAML [13] which selectively updates model parameters by gradient descents.
- **MetaCS** [27]: a gradient-based meta learning method similar to MeLU, except that it updates all parameters during meta-learning. As the codes of MetaCS are not available, we implement MetaCS based on the codes of MeLU.
- **MetaHIN**[6] [10], a method incorporating heterogeneous information networks into MAML to capture rich semantics from meta-paths.
- **MAMO**[7] [9]: a method also follows MAML but the model parameters are modulated by memory-augmented attention mechanism before local-update.
- **TaNP**[8] [15]: a hypernetwork-based method which learns to map item interaction records to modulate item- specific parameters.
- **ColdNAS**[9] [30]: a hypernetwork-based method which additionally uses neural architecture search to find the proper modulation function and modulation position.

[1]https://grouplens.org/datasets/movielens/1m/
[2]http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[3]https://grouplens.org/datasets/hetrec-2011/

[4]https://github.com/layer6ai-labs/DropoutNet
[5]https://github.com/hoyeoplee/MeLU
[6]https://github.com/rootlu/MetaHIN
[7]https://github.com/dongmanqing/Code-for-MAMO
[8]https://github.com/IIEdm/TaNP
[9]https://github.com/lars-research/coldnas

TABLE II
TEST PERFORMANCE (%) OBTAINED ON BENCHMARK DATASETS. THE BEST RESULTS ARE BOLDED AND THE SECOND-BEST RESULTS ARE UNDERLINED. FOR MSE AND MAE, SMALLER VALUE IS BETTER. FOR nDCG$_3$ AND nDCG$_5$, LARGER VALUE IS BETTER.

Dataset	Metric	DropoutNet	MeLU	MetaCS	MetaHIN	MAMO	TaNP	ColdNAS	ColdU
MovieLens	MSE	100.90$_{(0.70)}$	95.02$_{(0.03)}$	95.05$_{(0.04)}$	91.89$_{(0.06)}$	90.20$_{(0.22)}$	89.11$_{(0.18)}$	$\underline{87.96}_{(0.12)}$	**87.44**$_{(0.16)}$
	MAE	85.71$_{(0.48)}$	77.38$_{(0.25)}$	77.42$_{(0.26)}$	75.79$_{(0.27)}$	75.34$_{(0.26)}$	74.78$_{(0.14)}$	$\underline{74.29}_{(0.20)}$	**74.10**$_{(0.19)}$
	nDCG$_3$	69.21$_{(0.76)}$	74.43$_{(0.59)}$	74.46$_{(0.78)}$	74.69$_{(0.32)}$	74.95$_{(0.13)}$	75.60$_{(0.07)}$	$\underline{76.16}_{(0.03)}$	**76.21**$_{(0.07)}$
	nDCG$_5$	68.43$_{(0.48)}$	73.52$_{(0.41)}$	73.45$_{(0.56)}$	73.63$_{(0.22)}$	73.84$_{(0.16)}$	74.29$_{(0.12)}$	$\underline{74.74}_{(0.09)}$	**74.83**$_{(0.21)}$
BookCrossing	MSE	15.38$_{(0.23)}$	15.15$_{(0.02)}$	15.20$_{(0.08)}$	14.76$_{(0.07)}$	14.82$_{(0.05)}$	14.75$_{(0.05)}$	$\underline{14.15}_{(0.08)}$	**14.03**$_{(0.08)}$
	MAE	3.75$_{(0.01)}$	3.68$_{(0.01)}$	3.66$_{(0.01)}$	3.50$_{(0.01)}$	3.51$_{(0.02)}$	3.48$_{(0.01)}$	$\underline{3.40}_{(0.01)}$	**3.34**$_{(0.02)}$
	nDCG$_3$	77.66$_{(0.18)}$	77.69$_{(0.15)}$	77.68$_{(0.12)}$	77.66$_{(0.19)}$	77.68$_{(0.09)}$	77.48$_{(0.06)}$	$\underline{77.83}_{(0.01)}$	**77.88**$_{(0.04)}$
	nDCG$_5$	80.87$_{(0.15)}$	81.10$_{(0.15)}$	80.97$_{(0.09)}$	80.95$_{(0.04)}$	81.01$_{(0.05)}$	81.16$_{(0.21)}$	$\underline{81.32}_{(0.10)}$	**81.45**$_{(0.14)}$
Last.fm	MSE	21.91$_{(0.38)}$	21.69$_{(0.34)}$	21.68$_{(0.12)}$	21.43$_{(0.23)}$	21.64$_{(0.10)}$	21.58$_{(0.20)}$	$\underline{20.91}_{(0.05)}$	**20.85**$_{(0.06)}$
	MAE	43.02$_{(0.52)}$	42.28$_{(1.21)}$	42.28$_{(0.76)}$	42.07$_{(0.49)}$	42.30$_{(0.28)}$	42.15$_{(0.56)}$	$\underline{41.78}_{(0.24)}$	**41.78**$_{(0.19)}$
	nDCG$_3$	75.13$_{(0.48)}$	80.15$_{(2.09)}$	80.81$_{(0.97)}$	82.01$_{(0.56)}$	80.73$_{(0.80)}$	81.03$_{(0.36)}$	$\underline{82.80}_{(0.69)}$	**81.87**$_{(0.25)}$
	nDCG$_5$	69.03$_{(0.31)}$	75.03$_{(0.68)}$	75.01$_{(0.64)}$	75.98$_{(0.33)}$	75.45$_{(0.29)}$	75.98$_{(0.41)}$	$\underline{76.77}_{(0.10)}$	**76.83**$_{(0.12)}$



(a) MovieLens. (b) BookCrossing. (c) Last.fm.

Fig. 2. Varying number of layers of the predictor in ColdU.

Hyperparameter Setting: We find hyperparameters using the $\mathcal{T}^{\text{val}}$ via grid search. In ColdU, the batch size is 32. We choose a 4-layer predictor. The dimension of hidden units is set as $\boldsymbol{h}^1 = 128, \boldsymbol{h}^2 = 64, \boldsymbol{h}^3 = 32$. The learning rate is chosen from $\{5\times10^{-6}, 1\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}\}$ and the dimension of $\boldsymbol{r}_{i,j}$ in (2) is chosen from $\{128, 256, 512, 1024\}$.

Evaluation Metric: The performance is evaluated by mean average error (MAE), mean squared Error (MSE) which evaluate the numerical gap between the prediction and the ground-truth rating, and normalized discounted cumulative gain nDCG$_3$ and nDCG$_5$ which represent the proportion between the discounted cumulative gain of the predicted item list and the ground-truth list. For MAE and MSE , lower value is better. For nDCG$_3$ and nDCG$_5$, the higher value is better.

C. Performance Comparison

In Table II, we present the comprehensive user-cold start recommendation performance across all methods. Notably, ColdU exhibits better performance over other approaches across all datasets and metrics.

Among the compared baselines, DropoutNet performs the least favorably due to its non-few-shot learning nature, lacking the ability to adapt to diverse users. Within the realm of meta-learning based methods, MeLU, MetaCS, MetaHIN, and MAMO utilize a gradient-based meta-learning strategy, potentially susceptible to overfitting during local updates.

In contrast, TaNP and ColdNAS employ approaches that involve learning to generate user-specific parameters for guiding adaptation. TaNP employs a fixed modulation structure, which may not be optimally suited for different datasets. In contrast, ColdNAS automatically discovers an effective structure. Importantly, ColdU leverages the power of MLP to approximate complex functions. In particular, the use of MLP allows for the flexible modulation of the predictor in a layer-wise manner, enabling adaptability to the intricate patterns present in diverse cold-start user scenarios.

D. Effect of the Depth of Predictor

Finally, we analyze the effect of the depth L of predictor in ColdU. Figure 2 plots results obtained on MovieLens, BookCrossing and Last.fm.

As can be seen, the impact of choosing different values for L within a certain range reveals a low influence on performance. $L = 4$ already yields exceptional results as demonstrated in Table II. Thus, we choose $L = 4$ for simplicity.

V. Conclusion

In conclusion, this paper sheds light on the persistent challenge of user cold-start recommendation in recommendation systems. Introducing ColdU, our proposed approach leverages user-specific modulation through MLPs, achieving a balance between flexibility and parameter efficiency. Extensive experiments on benchmark datasets demonstrate ColdU consistently outperforms existing methods, establishing its prowess in addressing the user cold-start problem and contributing to the ongoing evolution of recommendation algorithms. The findings affirm the importance of personalized recommendation strategies for users with minimal interaction histories, showcasing the potential for advancements in the field.

References

[1] P. Resnick and H. R. Varian, “Recommender systems,” *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[2] F. M. Harper and J. A. Konstan, “The MovieLens datasets: History and context,” *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, 2015.

[3] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, “Improving recommendation lists through topic diversification,” in *International Conference on World Wide Web*, 2005, pp. 22–32.

[4] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, “Methods and metrics for cold-start recommendations,” in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002, pp. 253–260.

[5] Y.-J. Park and A. Tuzhilin, “The long tail of recommender systems and how to leverage it,” in *ACM Conference on Recommender Systems*, 2008, pp. 11–18.

[6] M. Volkovs, G. Yu, and T. Poutanen, “DropoutNet: Addressing cold start in recommender systems,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 4957–4966.

[7] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–34, 2020.

[8] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, “MeLU: Meta-learned user preference estimator for cold-start recommendation,” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1073–1082.

[9] M. Dong, F. Yuan, L. Yao, X. Xu, and L. Zhu, “MAMO: Memory-augmented meta-optimization for cold-start recommendation,” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 688–697.

[10] Y. Lu, Y. Fang, and C. Shi, “Meta-learning on heterogeneous information networks for cold-start recommendation,” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1563–1573.

[11] 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 *AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 10 772–10 780.

[12] L. Wang, B. Jin, Z. Huang, H. Zhao, D. Lian, Q. Liu, and E. Chen, “Preference-adaptive meta-learning for cold-start recommendation.” in *International Joint Conference on Artificial Intelligence*, 2021, pp. 1607–1614.

[13] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning*, 2017, pp. 1126–1135.

[14] D. Ha, A. Dai, and Q. V. Le, “Hypernetworks,” in *International Conference on Learning Representations*, 2017.

[15] X. Lin, J. Wu, C. Zhou, S. Pan, Y. Cao, and B. Wang, “Task-adaptive neural process for user cold-start recommendation,” in *The Web Conference*, 2021, pp. 1306–1316.

[16] X. Feng, C. Chen, D. Li, M. Zhao, J. Hao, and J. Wang, “CMML: Contextual modulation meta learning for cold-start recommendation,” in *ACM International Conference on Information and Knowledge Management*, 2021, pp. 484–493.

[17] H. Pang, F. Giunchiglia, X. Li, R. Guan, and X. Feng, “PNMTA: A pretrained network modulation and task adaptation approach for user cold-start recommendation,” in *The Web Conference*, 2022, pp. 348–359.

[18] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “FiLM: Visual reasoning with a general conditioning layer,” in *AAAI Conference on Artificial Intelligence*, vol. 32, 2018, pp. 3942–3951.

[19] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, “Fast and flexible multi-task classification using conditional neural adaptive processes,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 7957–7968.

[20] M. Brockschmidt, “GNN-FiLM: Graph neural networks with feature-wise linear modulation,” in *International Conference on Machine Learning*, 2020, pp. 1144–1152.

[21] C. Gao, Y. Li, Q. Yao, D. Jin, and Y. Li, “Progressive feature interaction search for deep sparse network,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 392–403.

[22] Y. Xie, Z. Wang, Y. Li, B. Ding, N. M. Gürel, C. Zhang, M. Huang, W. Lin, and J. Zhou, “FIVES: Feature interaction via edge search for large-scale tabular data,” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 3795–3805.

[23] Y. Koren, “Factorization meets the neighborhood: A multifaceted collaborative filtering model,” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2008, pp. 426–434.

[24] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, “AutoRec: Autoencoders meet collaborative filtering,” in *International Conference on World Wide Web*, 2015, pp. 111–112.

[25] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *International Conference on World Wide Web*, 2017, pp. 173–182.

[26] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua, “Addressing cold-start in app recommendation: latent user models constructed from twitter followers,” in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2013, pp. 283–292.

[27] H. Bharadhwaj, “Meta-learning for user cold-start recommendation,” in *International Joint Conference on Neural Networks*, 2019, pp. 1–8.

[28] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, “Wide & deep learning for recommender systems,” in *Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10.

[29] A. Pinkus, “Approximation theory of the MLP model in neural networks,” *Acta numerica*, vol. 8, pp. 143–195, 1999.

[30] S. Wu, Y. Wang, Q. Jing, D. Dong, D. Dou, and Q. Yao, “ColdNAS: Search to modulate for user cold-start recommendation,” in *The Web Conference*, 2023, pp. 1021–1031.