Online Item Cold-Start Recommendation with Popularity-Aware Meta-Learning

Anonymous Author(s)

ABSTRACT

With the rise of e-commerce and short videos, online recommender systems that can capture users' interests and update new items in real-time play an increasingly important role. In both online and offline recommendation, the cold-start problem due to interaction sparsity has been affecting the recommendation effect of cold-start items, which is also known as the long-tail problem of item distribution. Many cold-start scheme based on fine-tuning or knowledge transferring shows excellent performance on offline recommendation. Yet, these schemes are infeasible for online recommendation on streaming data pipelines due to different training method, computational overhead and time constraints.

Inspired by the above questions, we propose a model-agnostic recommendation algorithm called Popularity-Aware Meta-learning (PAM), to address the item cold-start problem under streaming data settings. PAM divides the incoming data into different meta-learning tasks by predefined item popularity thresholds. The model can distinguish and reweight behavior-related features and content-related features in each task based on their different roles in different popularity levels, thus adapting to recommendations for cold-start samples. These task-fixing design significantly reduces additional computation and storage costs compared to offline methods. Furthermore, PAM also introduced data augmentation and an additional self-supervised loss specifically designed for low-popularity tasks, leveraging insights from high-popularity samples. This approach effectively mitigates the issue of inadequate supervision due to the scarcity of cold-start samples. Experimental results across multiple public datasets demonstrate the superiority of our approach over other baseline methods in addressing cold-start challenges in online streaming data scenarios.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender System, Cold-Start Problem, Online Recommendation, Meta-Learning

1 INTRODUCTION

Platforms that depends on online recommendation service, including short video, e-commerce, and streaming media play an indispensable role in people's lives. Online recommender systems need to capture interest shifts in the system through periodic updates while taking into account the historical information of users and items. Many previous works [4, 32, 40] have demonstrated superior performance in the realm of online recommendation. These online systems are characterized by a large amount of data, strict time-consuming constraints, changes in global information (*e.g.* user interests, item popularity) over time [24] and the requirement of one epoch streaming training. A common way to train the system streamingly is continuously optimizing the model to capture information shifts. The same network parameters are applied for all users and items, and the parameters are updated using data arriving during the period to avoid being stale.

Another inevitable recommendation scenario is the cold-start problem caused by sparse consumption data of new users or items. The fact that low-popularity items comprise a huge portion of the total items and their interaction data represents only a small fraction of the overall consumption data [8, 9] is also concerning. As a consequence of this long-tail distribution, the model excels in recommending popular items but struggles when it comes to recommending cold items. On the flip side, interactions with popular items provide a more accurate reflection of user interests and item features. Even recommendations for cold-start items heavily rely on the patterns learned by the model from these interactions.

Such cold-start problem has garnered considerable attention and has been addressed through various solutions in recommendation scenarios. The representation-based approach [12, 15, 42] considers items as tasks and generates personalized parameters for different items to enhance the effectiveness of cold-start, such as meta-learning [15] and knowledge transfer [42]. The side-information-based approach [21, 39] focuses on using side information and pay less attention to the item's popularity-related information, which also have different importance in the recommendation of cold and popular items. However, the methods are still challenging to apply in online scenarios with streaming data.

The cold-start problem in online recommendations faces two main challenges. On the one hand, in the combination of online and cold-start recommendation, the Matthew effect [19] makes the system optimize the distribution of popular items further. Because the popular items make up the majority of the training data leading to better performance on popular items, thus the data further increases the proportion of popular data over time. This effect exacerbates the problem of cold-starts in online systems. Meanwhile, the requirement of training streaming data with one epoch makes the existing cold-start methods inapplicable, and the time consumption and computational demand make generating personalized parameters for items impractical in online scenarios.

In this work, we have introduced a novel model-agnostic **P**opularity-**A**ware**d** **M**eta-learning framework (PAM) for solving the item cold-start problem under streaming data in online systems. Firstly, we propose a fixed task segmentation with predefined popularity thresholds in the gradient-based meta-learning [7]. We leverage meta-learning's ability to share meta-parameters between tasks and generate personalized parameters for different tasks. In this way, we optimise the performance of cold-start tasks without losing the interest information of streaming data, and the personalized parameters also avoid making the system over-fitting popular item data, solving the Matthew cold-start problem in online scenarios. In PAM, tasks of varying popularity levels can share meta-knowledge

and utilize popularity-related features and content-related features with different weights. Additionally, the fixed task segmentation enables efficient online inference without fine-tuning on each item, thus adapts to streaming training scenarios while addressing the computational overhead and time constraints.

Besides, we also design a cold-start task enhancer to further utilize various information from popular items in the system to optimize the cold-start task. The enhancer classifies the embeddings into behavior-based and content-based (because of their different roles in the recommendation of cold-start and popular items) and comprises two auxiliary tasks based on these two types of embeddings: 1) The data augmentation module constructs low-popularity samples using high-popularity interaction data, thereby directly increasing the number of cold-start samples; 2) the self-supervised module replay historical cold-start interactions leveraging well-trained item embeddings as supervision, thereby promoting the feature extraction capability for cold-start tasks.

Our main contributions can be summarized as follows:

- We proposed a meta-learning algorithm that effectively tackles the item cold-start problem in online recommendation without requiring additional fine-tuning in online serving (Sec. 4.4.2).
- We introduced a unique cold-start enhancer that effectively mitigates the issue of scarcity feedback for cold-start items (Sec. 4.3).
- We conducted complete experiments on three benchmark datasets and a real online recommendation scenario. And our method surpasses previous baselines by a large margin.

2 RELATED WORKS

In this section, we discuss researches relevant to this work, including meta-learning, cold-start and online recommendation.

2.1 Cold-Start Recommendation

Solutions to the cold-start problem are usually categorized into side-information-based methods and fine-tuning-based methods [17, 22]. Cold-start recommendation methods based on side information are diverse [17, 21, 25, 27, 31]. The more side-information is available to the system in online recommendation, the better the performance of such methods. Approaches based on fine-tuning are also suitable for cold-start scenarios, where features are extracted in advance through pre-training or meta-knowledge and can be quickly adapted to the task to achieve better results when faced with a new scenario with fewer samples [12]. In recent years, many hybrid methods that introduce side-information into few-shot learning have also emerged. As a method that utilizes a meta-learning framework to introduce content information to make recommendations for items of different popularity, PAM is likewise a hybrid method that combines both. Meta-learning is also an efficient solution to adapt quickly to personalized parameters with a few samples, and we will describe the various methods in the subsequent subsections.

2.2 Online Recommendation

The goal of online recommendation is to assist users in discovering items they may be interested in real-time, and be able to distribute new items in the system promptly. Unlike sequential recommendations, which focus on the historical behavior of the system, online recommendations also have to take into account real-time interest shifts and be able to generate recommendation parameters in a low time-consuming process. As mentioned earlier, the large amount of online data and the streaming training method leads to many schemes that generate specialized parameters for sequential recommendation in offline systems to be infeasible. There are many types of common approaches, including CTR prediction based IncCTR [32] and DDP [38], and FIRE [36] that proposes a incremental recommendation algorithm from the perspective of graph signal processing. Various types of meta-learning algorithms introduced in the next subsection also shows potential in online recommendations. However, they still struggle with the long-tail distribution of online data and performance poorly on cold-start items.

2.3 Meta-Learning

Meta-learning [7], also known as *learning to learn*, is an emerging few-shot learning method that seeks to quickly adapt to the corresponding task with a few-shot data to achieve better recommendation results. The meta-learning training process divide the training set into support and query set, the former are used to generate unique parameters and the latter are used to calculate loss for training. Since MeLU [15], meta-learning has also demonstrated its applicability in the field of recommender systems due to the high overlap between few-shot learning and cold-start recommendation scenarios [3, 5, 15, 16, 18, 20, 23, 30, 35, 41]. In recent work, ClusterSeq [18] preserves secondary user preferences through a sequential recommendation system based on meta-learning clustering. Online recommendation is also another scenario where the training process of meta-learning is applicable because of its characteristic of making recommendation model updates in a temporal order [6, 13, 24, 28, 37, 40]. As an example, SML [40] trains a meta-model to output new parameters based on the parameters of the past moments and the data of the current moment. FORM [28] adapts user-specific tasks and leverages the online meta leader to optimize online training performance. While FORM alleviates the time cost of fine tuning for individual users, it continues to face challenges in meeting the real-time demands in streaming data scenarios, such as product or video recommendations.

3 PRELIMINARY

3.1 Notations

In this work, we consider item cold-start recommendation task in streaming data scenario. The data arrives by time period $\{D_0, D_1, \cdots, D_t, \cdots\}$, each period of data D_t consists of user-item interactions $(u, i) \in \mathcal{U} \times \mathcal{I}$ that happened during D_t. The recommender system is similarly updated by time, with $\Phi_t, \Theta_t, \Omega_t$ representing different parts recommender at time t. The system maintains the embedding matrix parameters Φ and the network weight parameters Ω. We also define the initialization of the network weights as Θ. Matrices and vectors are denoted by symbols in bold font. For a particular interaction data (u, i), $v_i \in \mathbb{N}$ represents the number of views of the item, *i.e.*, the number of times the item has been clicked by the user prior to this moment in time. In particular, for some pre-given threshold v_{cold}, items with popularity below that threshold are called cold-start items, and others are called popular items.

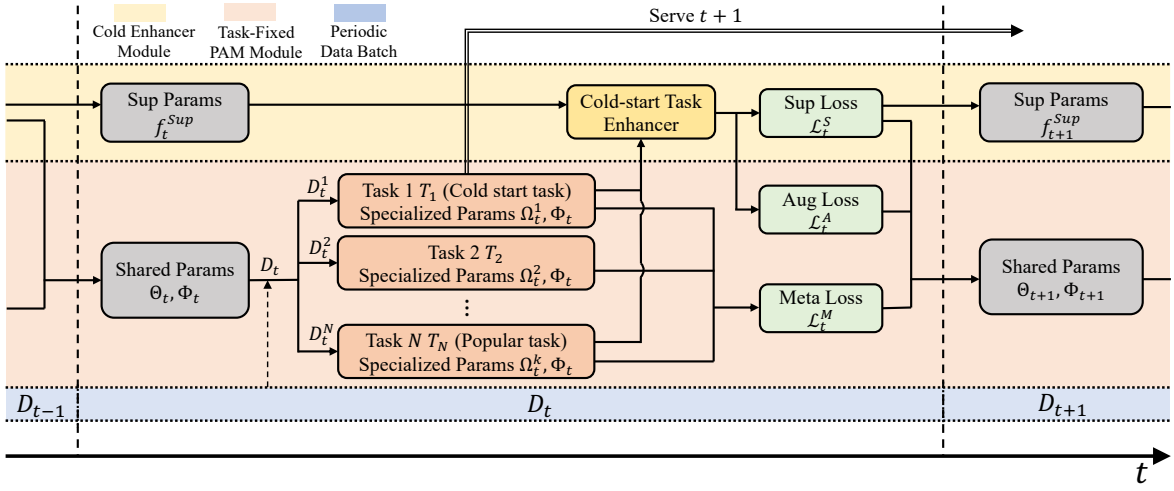


Figure 1: The overview of the proposed PAM method. Φ_t denotes the task-shared embedding parameters, Θ_t denotes the task-shared network parameters initialization that will be fine-tuned to specialized parameters Ω_t^n using D_t^n for each task.

3.2 Problem Statement

Our goal is to consider the item cold-start problem in online recommender systems. In traditional online recommendation schemes, the system periodically retrains the system. To ensure that various types of historical data in the system are preserved and the real-time interest is captured, the generation of the current recommendation system W_t needs to encapsulate the past information and search for better performance on the preceding data period:

$$D_{t+1} \Leftarrow W_t \leftarrow \mathcal{M}(W_{t-1}, \{D_i, i \leq t\}) \tag{1}$$

where $\mathcal{M}$ can be a gradient minimization or a meta generator, *e.g.*, MeLON [13] and SML [40]. $\Leftarrow$ stands for serving. However, the above approaches suffer from the poor performance on cold-start items brought about by the scarcity of cold-start data. The cold-start online recommendation problem can be formulated as:

$$D_{t+1}^{cold} \Leftarrow W_t \leftarrow \mathcal{M}(W_{t-1}, \{D_i, i \leq t\}) \tag{2}$$

where D^{cold} stands for the cold-start item part of data. Next, we present the PAM method that addresses the problem of cold-start items in online recommender systems.

4 METHOD

In this section, we describe the proposed training method in detail, including the task-fixed meta-learning module, cold-start instruction module, and the base recommender model we adopt. The overview of the PAM method is described in Fig. 1.

4.1 Base Recommender Model

We would highlight that PAM is a model-agnostic online training approach. We utilize a dual-tower structure [11] to ensure that it is uniform across all frameworks in this paper.

4.1.1 Embedding layer. $(u, i) \rightarrow \boldsymbol{e}_u, \boldsymbol{e}_i$: The purpose of the embedding layer is to characterize the different information about the user and the item, convert it into a vector, and output it to the hidden layer for information extraction. Specifically, the embedding layer consists of multiple feature embedding matrices of users and items. For user u, the system maintains multiple one-hot vectors for different features, which extract the user's information from the feature embedding matrix and then concatenate it into the embedding vector:

$$\boldsymbol{e}_u = [\boldsymbol{e}_{u1}, \cdots, \boldsymbol{e}_{uP}] = [\boldsymbol{E}_{U1}\boldsymbol{c}_{u1}, \cdots, \boldsymbol{E}_{UP}\boldsymbol{c}_{uP}] \tag{3}$$

where P is the number of user features, $\boldsymbol{E}_{Up} \in \Phi_t$ represents the p-th feature embedding matrix and $\boldsymbol{c}_{up}$ stands for the one-hot vector for user feature p. Items have similar embedding outputs as users:

$$\boldsymbol{e}_i = [\boldsymbol{e}_{i1}, \cdots, \boldsymbol{e}_{iQ}] = [\boldsymbol{E}_{I1}\boldsymbol{c}_{i1}, \cdots, \boldsymbol{E}_{IQ}\boldsymbol{c}_{iQ}] \tag{4}$$

4.1.2 Hidden layer. $\boldsymbol{e}_u, \boldsymbol{e}_i \rightarrow \boldsymbol{z}_u, \boldsymbol{z}_i$: After the embedding vectors are generated, they are fed into the hidden layer and ultimately output the top representations of the user and item. In our dual-tower model, the user and the item have their own fully connected layer parameters, and the embedding vector will be mapped to the top representation by L fully connected network layers $\boldsymbol{z}_u = f_{uL} \circ \cdots \circ f_{u1}(\boldsymbol{e}_u)$ and $\boldsymbol{z}_i = f_{iL} \circ \cdots \circ f_{i1}(\boldsymbol{e}_i)$, where the l-th layer can be represented as:

$$f_l(\boldsymbol{e}) = \sigma(\boldsymbol{W}_l \boldsymbol{e} + \boldsymbol{b}_l) \tag{5}$$

where $\boldsymbol{W}_l, \boldsymbol{b}_l \in \Omega_t$ are the weight matrix and bias vector of the l-th hidden layer, and σ represents a ReLU [1] activation function.

4.1.3 Output layer. $\boldsymbol{z}_u, \boldsymbol{z}_i \rightarrow \hat{y}_{ui}$: The output layer calculates the predicted scores for the corresponding user-item pairs based on the top-level representations of the users and items. We utilize the InfoNCE [29] loss form to calculate the prediction and loss value. The prediction of interaction $\hat{y_{ui}}$ is calculated by sampling other interactions in the batch as negative samples:

$$\hat{y_{ui}} = \frac{\exp(\boldsymbol{z}_u \cdot \boldsymbol{z}_i / \tau)}{\sum_{u' \in D_t} \exp(\boldsymbol{z}_{u'} \cdot \boldsymbol{z}_i / \tau)} \tag{6}$$

where τ is the temperature hyperparameter in InfoNCE loss to control the model's discrimination for negative samples. The log

loss form is adopted to calculate the gradients:

$$\mathcal{L}_T = -y_{ui} \cdot \log \hat{y_{ui}} - (1 - y_{ui}) \cdot \log(1 - \hat{y_{ui}}) \tag{7}$$

where $\hat{y_{ui}}$ and y_{ui} refer to the prediction and label values of (u, i).

4.2 Popularity-Aware Meta-Learning

As the cold-start item samples constitute only a small fraction of the online traffic, their influence on model updating is limited. Consequently, the recommender system may exhibit satisfactory performance for popular items but may perform poorly for cold-start items. However, directly excluding samples from popular items could result in reduced performance for both cold-start and popular items. This is because samples from popular items also contain user interest-related information, and the exclusion leads to an information loss.

Hence, it naturally occurred to us to partition the tasks according to popularity and employ gradient-based meta-learning for recommendation. This popularity-aware segmentation enables us to strike a balance by avoiding excessive emphasis on highly popular items while still leveraging the valuable information from interactions with high-popularity items. Meanwhile, the segmentation is fixed during online training and it offers benefits from two perspectives. Firstly, it mitigates the overhead associated with traditional meta-learning patterns, which treat every data instance as a separate task, thus requiring task-specific parameter updating and storage. These methods incur unacceptable costs for online application. Secondly, as mentioned in [26], the meta-learner extracts versatile features for all tasks, and after fine-tuning on specific task, the model can reuse these features with different weights for fast adaption. In PAM, items with the same popularity are recommended with identical features, while tasks with different popularity share lower-level embedding features. In other words, cold-start items can leverage more popularity-independent features, *i.e.*, content features, while popular items can rely more on popularity-related features, *i.e.*, historical feedback features.

4.2.1 Fixed Task Segmentation. Specifically, for a designated number of tasks N and an interaction (u, i), the number of tasks to which this interaction belongs is calculated by a piece-wise constant function:

$$f(v_i) : v_i \in \mathbb{N} \rightarrow T_n \in \{T_1, \cdots, T_N\} \tag{8}$$

where T_n represents the n-th task, and v_i is a metric reflects popularity, such as the number of click or sales volume. The function f is defined by a set of pre-determined thresholds.

4.2.2 Local Updates. The parameters in PAM follow a bi-level optimization scheme, consisting of local updates and global updates. Before the training process starts, we initialize the global recommender parameters $\boldsymbol{E}_{Up}, \boldsymbol{E}_{Iq} \in \Phi_t, \boldsymbol{W}_l, \boldsymbol{b}_l \in \Theta_t$. After the arrival of a data batch D_t, the batch will be divided into multiple parts $\{D_t^1, \cdots, D_t^N\}$ according to the popularity of the item by function f. Each portion of data D_t^n is further divided into D_t^{nS} and D_t^{nQ}, standing for support and query set. For the n-th task, the personalized recommender parameters Ω_t^n can be obtained by local updates:

$$\Omega_t^n \leftarrow \Theta_t - \alpha \nabla_{\Theta_t} \mathcal{L}_{T_n}(\Theta_t, \Phi_t^n | D_t^{nS}) \tag{9}$$

where α represents the inner loop learning rate, and $\mathcal{L}_T(\Theta|D)$ represents the loss function of task T on data set D with an initialization of Θ (see Eq. (7)). The recommender $\{\Phi_t, \Omega_t^n\}$ will be used to serve the items in task n at the following time moment $t+1$. Note that the interaction data between the support set and the query set have no overlap, hence it is meaningless to update the embedding parameters of users and items in the local update, and the updated parameters contain only weight parameters of the networks in Eq. (9). We utilize the LSLR [2] methods, maintaining an adaptive learning rate for each network weight to alleviate overfitting and advance the performance.

4.2.3 Global Updates. The goal of the meta optimization is to minimize the sum of the losses of the query set D_t^{nQ} over its parameters Ω_t^n in each task T_n. Since all of the parameters are involved in the computation of the target function, the global update will be performed on all of the parameters:

$$\{\Phi_{t+1}, \Theta_{t+1}\} \leftarrow \{\Phi_t, \Theta_t\} - \beta \nabla_{\{\Phi_t, \Theta_t\}} \mathcal{L}_t^M \tag{10}$$

where $\mathcal{L}_t^M$ is the main meta-learning loss, β stands for the outer loop learning rate, and

$$\mathcal{L}_t^M = \sum_{n=1}^{N} \lambda_n \mathcal{L}_{T_n}(\Omega_t^n, \Phi_t^n | D_t^{nQ}) \tag{11}$$

where λ represents the personalized weight of tasks. Because of the fixed division of the tasks, we can leverage the importance of the task and assign distinctive weights. For instance, the cold-start task, involving less popular items, can be assigned a higher weight.

By fixed tasks division as described above, we solve the problem of long-tailed distribution of training data by dividing the cold-started long-tailed part of the item distribution into separate tasks.

4.3 Cold-start Task Enhancer

Apart from task partitioning in meta-learning, incorporating additional supervision signals can prevent the neglect of cold-start tasks during online training. Thus, we devise a cold-start task enhancer that effectively transfers information from popular items and their associated interaction feedback. This module further improve the recommendations in the cold-start task.

4.3.1 Cold-start Embedding Simulation. We categorize the various types of feature embedding of items into two types: behavior-based embeddings $\boldsymbol{e}_q^{beh}$ and content-based embeddings $\boldsymbol{e}_q^{con}$, because these types of embeddings play different roles in the recommendation of cold-start and popular items, as shown in Sec. 5.4. We then further divide the behavior-based embeddings into ID embedding and sequential embeddings, this is because that ID embedding directly stores the information of the items and we utilize it for the cold-start embedding simulation, the details of which will be described later. The system maintains a cold-start embedding parameter $\hat{\Phi}$, for each item i, whenever it appears in the data D_{t-k}^{cold} as a cold-start item, the system stores its behavior-based embeddings (ID embedding and sequential embeddings) into the parameter $\hat{\Phi}_{t-k}$ in addition to updating its own embedding parameter Φ_{t-k}. For each item i that appears in the popular task D_{t-k}^{hot}, its behavior-based embeddings in the cold-start period have certainly been stored. Therefore, we can simulate a cold-started item at the

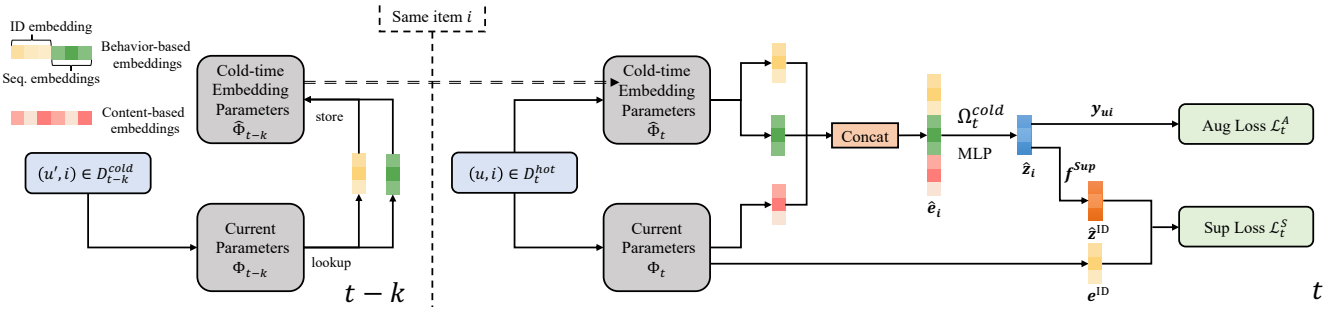


Figure 2: The structure of our proposed cold-start task enhancer.

current moment with the help of the stored embedding $\hat{\boldsymbol{e}}_q^{beh} \in \hat{\Phi}_{t-k}$ as well as other content-based feature embedding $\boldsymbol{e}_q^{con} \in \Phi_t$:

$$\hat{\boldsymbol{e}}_i = [\hat{\boldsymbol{e}}_{i1}^{beh}, \cdots, \hat{\boldsymbol{e}}_{ik}^{beh}, \boldsymbol{e}_{ik+1}^{con}, \cdots, \boldsymbol{e}_{iQ}^{con}] \tag{12}$$

The equivalent write-up for dividing behavior-based embeddings $\hat{\boldsymbol{e}}_q^{beh}$ into ID embedding $\hat{\boldsymbol{e}}_q^{ID}$ and sequential embeddings $\hat{\boldsymbol{e}}_q^{seq}$ is:

$$\hat{\boldsymbol{e}}_i = [\hat{\boldsymbol{e}}_i^{ID}, \hat{\boldsymbol{e}}_{i2}^{seq}, \cdots, \hat{\boldsymbol{e}}_{ik}^{seq}, \boldsymbol{e}_{ik+1}^{con}, \cdots, \boldsymbol{e}_{iQ}^{con}] \tag{13}$$

and $\hat{\boldsymbol{e}}_i$ fully simulates a cold-start item embedding at current time.

4.3.2 Data Augmentation. With relatively low traffic proportion, the cold-start task in meta-learning still has limited training opportunity. Data argumentation can significant increase the number of cold-start samples, so we propose a simulation-based data argumentation. We treat the simulated cold-start embedding $\hat{\boldsymbol{e}}_i$ from the previous subsection as a new part of cold-start task data with the real popular interaction data labels y_{ui}, and train the meta-learning model to update the cold-start task parameters, which can partially augment the data to improve the effect.

We denote the simulated cold-start interaction data as $\hat{D}_t^{cold}$ and similarly divide them into support and query sets $\hat{D}_t^{cold,S}$, $\hat{D}_t^{cold,Q}$. Similar to Eq. (10), the loss function form of data augmentation is computed from the data in the support set:

$$\mathcal{L}_t^A = \mathcal{L}(\hat{\Omega}_t^{cold} | \hat{D}_t^{cold,Q}) \tag{14}$$

where $\hat{\Omega}_t^{cold}$ is calculated by support sets $\hat{D}_t^{cold,S}$.

4.3.3 Self-supervised Instructor. Furthermore, we propose leveraging the well-learned embeddings of popular items and achieve instruction for the cold-start task by training a mapping network to fit the transformation process of the cold-start embedding to the embedding from popular phase. For the simulated cold-start item embedding $\hat{\boldsymbol{e}}_i$, we have the ID embedding of its popular state $\boldsymbol{e}^{\text{ID}} \in \Phi_t$, which has more information due to multiple updates. To this end, we reinforce the ability of the other fully connected layers to extract information by replacing the last fully connected layer of the cold-start task network parameters $\boldsymbol{W}_L, \boldsymbol{b}_L$ with a unique mapping parameter $\boldsymbol{W}^{Sup}, \boldsymbol{b}^{Sup}$ that makes the output top embedding as similar as possible to its true ID embedding:

$$\hat{\boldsymbol{z}}^{\text{ID}} = f^{Sup}(\boldsymbol{e}_{L-1}) = \boldsymbol{W}^{Sup}\boldsymbol{e}_{L-1} + \boldsymbol{b}^{Sup} \tag{15}$$

Algorithm 1 The training and test process of PAM.

Require: Hyperparameters $\alpha, \beta, \lambda, \gamma, f$
  Randomly initialize parameters $\Phi_0, \Theta_0, \alpha_0, f_0^{Sup}$
  while incoming periodic data D_t **do**
    $\{D_t^1, \cdots, D_t^N\} \leftarrow f(D_t)$ ▷ Divide data by popularity
    for $n \in \{1, \cdots, N\}$ **do**
      $D_t^{nS}, D_t^{nQ} \leftarrow D_t^n$
      $\Omega_t^n \leftarrow \Theta_t - \alpha \nabla_{\Theta_t} \mathcal{L}_{T_n}(\Theta_t, \Phi_t^n | D_t^{nS})$
    end for
    $\mathcal{L}_t^M = \sum_{n=1}^k \lambda_n \mathcal{L}_{T_n}(\Omega_t^n, \Phi_t^n | D_t^{nQ})$ ▷ Main meta-learning loss
    if $t \geq \tau$ **then**
      Store cold-start parameter Ω_t^{cold}
      Serve D_t^{cold} using Φ_{t-1}, Ω_{t-1}^{cold} ▷ Serving period
    end if
    Store Φ_t of cold-start part D_t^{cold} into $\hat{\Phi}_t$
    Concatenate $\hat{\boldsymbol{e}}_i$ using $\hat{\Phi}_t, \Phi_t$ of hot part D_t^{hot}
    Lookup $\boldsymbol{e}_i^{\text{ID}}$ using Φ_t of hot part D_t^{hot}
    Calculate $\hat{\boldsymbol{z}}^{\text{ID}}$ using $\hat{\boldsymbol{e}}_i$ and network parameters $\Omega_t^{cold}, f_t^{Sup}$
    $\mathcal{L}_t^S \leftarrow \frac{1}{N} \| \hat{\boldsymbol{z}}^{\text{ID}}, \boldsymbol{e}^{\text{ID}} \|_2^2$ ▷ Self-supervised instructor loss
    Form a simulated cold task $\hat{D}_t^{cold,S}, \hat{D}_t^{cold,Q}$ using $\hat{\boldsymbol{e}}_i, D_t^{hot}$
    $\hat{\Omega}_t^n \leftarrow \Theta_t - \nabla_{\Theta_t} \alpha_{\Theta_t} \mathcal{L}(\Theta_t, \hat{\Phi}_t^n | \hat{D}_t^{cold,S})$
    $\mathcal{L}_t^A \leftarrow \mathcal{L}(\hat{\Omega}_t^{cold} | \hat{D}_t^{cold,Q})$ ▷ Data augmentation loss
    $\mathcal{L}_t^T \leftarrow \gamma_M \mathcal{L}_t^M + \gamma_S \mathcal{L}_t^S + \gamma_A \mathcal{L}_t^A$ ▷ Final total loss
    $\{\Phi_{t+1}, \Theta_{t+1}, f_{t+1}^{Sup}\} \leftarrow \{\Phi_t, \Theta_t, f_t^{Sup}\} - \beta \nabla_{\{\Phi_t, \Theta_t, f_t^{Sup}\}} \mathcal{L}_t^T$
  end while

where $\hat{\boldsymbol{z}}^{\text{ID}}$ is the output embedding. We use MSE loss to evaluate the similarity between output top embedding and true ID embedding:

$$\mathcal{L}_t^S = \frac{1}{N} \| \hat{\boldsymbol{z}}^{\text{ID}}, \boldsymbol{e}^{\text{ID}} \|_2^2 \tag{16}$$

where N stands for the dimension of embedding.

With the instruction of the popular ID embedding, we can implement a self-supervised learning-like method without the use of labels to make the cold-start task network parameters have better performance in extracting information.

4.4 Training and Serving

4.4.1 Training Process. With the introduction of the parts of the cold-start task enhancer, the update method of the recommender

Table 1: Overview of the datasets.

Dataset	#Users	#Items	#Inter.	#Tags	#Sparsity
MovieLens	43,181	51,142	6,840,091	20	3.09‰
Yelp	1,987,929	150,346	6,990,280	1,312	0.02‰
Book	351,487	581,717	6,402,728	2,808	0.03‰

parameters in Eq. (10) changes. We form a new total loss function by combining the three components of main meta-learning loss, self-supervised learning loss, and data augmentation loss:

$$\mathcal{L}_t^T = \gamma_M \mathcal{L}_t^M + \gamma_S \mathcal{L}_t^S + \gamma_A \mathcal{L}_t^A \tag{17}$$

where γ refers to the different weights of losses, and the new parameter update method becomes:

$$\{\Phi_{t+1}, \Theta_{t+1}, f_{t+1}^{Sup}\} \leftarrow \{\Phi_t, \Theta_t, f_t^{Sup}\} - \beta \nabla_{\{\Phi_t, \Theta_t, f_t^{Sup}\}} \mathcal{L}_t^T. \tag{18}$$

4.4.2 Online Serving. During the training process, the model is fine-tuned on the cold-start data to generate parameters for the cold-start task Ω_t^{cold}. We save that parameter and serve it for the recommendation of cold-start items at the next moment. In this case, PAM is able to store the parameters prepared for the cold-start item in advance without real-time personalized fine-tuning, thus avoiding the time cost of online fine-tuning altogether. Compared to traditional meta-learning, we avoid the time-consuming serving of personalization on each item, and also reduce the storage space overhead of maintaining the respective parameters for each item.

The training and serving process of PAM is described by Alg. 1.

5 EXPERIMENTS

We conducted experiments to answer the following RQs: **RQ1**: How does PAM's performance on online cold-start recommendations improve over existing frameworks? **RQ2**: How much of a performance enhancement do the various components of PAM provide? **RQ3**: Why PAM perform better on multitasking? **RQ4**: How does PAM perform in online A/B testing? **RQ5**: How do the various hyperparameters of PAM affect the performance results? (Sec. A.2).

5.1 Experimental Settings

5.1.1 Datasets. We selected three public datasets that containing timestamp to ensure that the characteristic of online streaming data is simulated, including **MovieLens** [9], **Yelp** and **Book** [10]. The detailed information of the datasets are elaborated in Sec. A.1.1.

For the label processing of the datasets, the user ratings lie between 0 and 5, while we consider the interaction data with ratings above 3 as positive samples and the rest as negative samples. The information of datasets is conducted in Table 1.

5.1.2 Baselines. We compare our proposed PAM with the following baseline, including **Periodical Fine-tuning**, **s^{2}Meta [6]**, **IncCTR [32]**, **SML [40]**, **ASMG [24]**, **MeLON [13]**, **IMSR [34]**, where all scenarios leverage the previously mentioned two-tower model structure and the feature inputs remain the same. The detailed information are showed in Sec. A.1.2.

5.1.3 Evaluation Metrics. Similar to ASMG [24], we divide each dataset equally into 31 periods, and each period is further partitioned into batches for training. To simulate the characteristics of online streaming data, we do not use a pre-training scheme, and all models will be consistently streamed starting from the data of the first period. We start testing from the 24th period, and the models obtained from each training period will be tested on cold-start data from the next period. The average of the metrics from all the testing phases will be reported. For specific metrics, we use the Recall@K and NDCG@K [33] metrics to measure the model's effectiveness on cold-start items. Since ranking the item's scores for all users is time-consuming, for a cold-start item, we take the interacted user as 1 positive sample and all the non-interacted users in the current batch as the negative samples (not less than 900 due to the item's low popularity), and compute the above metrics.

5.1.4 Hyperparameters. For the definition of the tasks, to simulate an online cold-start scenario, we designated the items with the lowest 5% of popularity in the interaction data as cold-start items [5, 17]. The cold-start thresholds of popularity v_{cold} on the three datasets are 50, 20, and 15, respectively. We divide the remaining data into 4 tasks to distinguish popular items of different natures.

During the training process, we use Adam [14] as the optimizer for gradient descent with α set to 0.001. The outer loop learning rate β is set to 0.001. The batch size is 1024, the weights of tasks β are set to 2 for the cold-start task and 0.5 for the other tasks, and the weights of the loss function γ are set to 1, 3, and 2, respectively.

5.2 Overall Performance (RQ1)

5.2.1 Comparison of Cold Items. Table 2 demonstrates results of the top-K metrics on cold-start items for various methods. From the result of the experiment, we have the following observations.

Our proposed PAM significantly outperforms other baselines on cold-start items. This demonstrates the effectiveness of PAM's unique meta-learning scheme targeted at solving long-tailed distributions of item popularity in online streaming data scenarios. The PAM performs better on smaller K-value metrics compared to other baseline methods, and the improvement is relatively small as the K-value increases. This suggests that the PAM method can accurately recommend cold-start items to the users who favor it the most. In contrast, the baseline method ranks the positive-sample users relatively far down the list, as reflected in the considerable rise in the metrics as the K-value rises.

SML and ASMG methods focus on extracting information from historical recommender systems and balancing the weights of historical information with real-time interests to optimize recommendations at the next moment. However, because data distribution tends to be popular videos, focusing on historical information only aggravates the interest bias of the model. Besides, this model relies more on the efficacy of the initialized model, which results in poorer performance on cold-start items in the streaming no pre-training scheme. At the same time, ASMG has a high storage space overhead (leads to OOM on the Yelp dataset), which becomes another limitation for its application in online systems. IMSR is also an approach that balances periodic retraining of historical with current information, and thus has a similarly reliance on the efficiency of pre-training, and performs poorly in schemes without pre-training.

Table 2: The reported top-K evaluation metric results on cold-start items. Bold represents the optimal result, underlined represents the optimal result in baseline, and Impr. % represents the improvement compared to the optimal result in baseline.

Dataset	Metric	PF	s^2Meta	IncCTR	SML	ASMG	MeLON	IMSR	PAM-M	PAM-S	PAM-A	PAM-F	Impr. %
MovieLens	Recall@5	0.2581	0.2344	0.2685	0.1965	0.1945	0.3132	0.2115	0.3846	0.3968	0.4114	**0.4157**	+32.73%
	Recall@10	0.3268	0.2750	0.3383	0.2628	0.2575	0.3925	0.2885	0.4733	0.4856	0.4980	**0.5036**	+28.31%
	Recall@20	0.4121	0.3342	0.4221	0.3470	0.3404	0.4862	0.3846	0.5727	0.5840	0.5912	**0.5966**	+22.71%
	NDCG@5	0.2011	0.1967	0.2100	0.1448	0.1427	0.2009	0.1668	0.3040	0.3162	0.3266	**0.3314**	+57.81%
	NDCG@10	0.2233	0.2060	0.2326	0.1661	0.1630	0.2429	0.1769	0.3327	0.3450	0.3547	**0.3598**	+48.13%
	NDCG@20	0.2448	0.2180	0.2813	0.1873	0.1839	0.2704	0.1816	0.3578	0.3698	0.3782	**0.3833**	+36.26%
Yelp	Recall@5	0.1264	0.1375	0.1350	0.0852	OOM	0.1253	0.0918	0.1969	0.2184	0.2203	**0.2262**	+64.51%
	Recall@10	0.2106	0.2206	0.2247	0.1513	OOM	0.2090	0.1836	0.2966	0.3261	0.3244	**0.3316**	+47.57%
	Recall@20	0.3324	0.3390	0.3496	0.2574	OOM	0.3327	0.2755	0.4199	0.4512	0.4472	**0.4538**	+29.81%
	NDCG@5	0.0807	0.0876	0.0858	0.0528	OOM	0.0791	0.0631	0.1313	0.1466	0.1467	**0.1525**	+74.09%
	NDCG@10	0.1077	0.1129	0.1145	0.0740	OOM	0.1062	0.0911	0.1633	0.1813	0.1802	**0.1865**	+62.88%
	NDCG@20	0.1383	0.1416	0.1459	0.1006	OOM	0.1370	0.1140	0.1944	0.2129	0.2112	**0.2173**	+48.93%
Book	Recall@5	0.2131	0.2146	0.2173	0.2104	0.2182	0.2002	0.2117	0.2456	0.2561	0.2468	**0.2609**	+20.06%
	Recall@10	0.2991	0.3022	0.3048	0.2943	0.3066	0.3059	0.2090	0.3460	0.3583	0.3478	**0.3616**	+18.21%
	Recall@20	0.4033	0.4056	0.4079	0.3968	0.4126	0.4311	0.3327	0.4614	0.4753	0.4667	**0.4806**	+11.48%
	NDCG@5	0.1475	0.1495	0.1517	0.1461	0.1519	0.1334	0.1204	0.1714	0.1789	0.1715	**0.1824**	+20.23%
	NDCG@10	0.1752	0.1777	0.1799	0.1731	0.1804	0.1646	0.1546	0.2038	0.2119	0.2040	**0.2149**	+19.46%
	NDCG@20	0.2015	0.2038	0.2059	0.1990	0.2071	0.2170	0.2129	0.2329	0.2414	0.2340	**0.2449**	+12.86%

In addition, the emphasis on historical information extraction also leads to a worse performance on cold-start recommendations.

IncCTR, as a knowledge distillation scheme that utilizes the model's self-supervised signals, also references historical information. Its low dependence on initialized parameters leads to a better fit to online scenarios and a slight improvement over PF, achieving state-of-the-art for cold-start items on some datasets. However, its way of generating additional labels increases the computational time consumption and partially reduces the model's efficiency.

s^2Meta is a scheme for task segmentation and meta-learning training of arriving data by categories of items. However, it only improves slightly relative to the PF baseline when no new item categories enter the system and performs slightly inferior on the content-information-rich MovieLens dataset, as the rest of the baseline can similarly achieve similar information extraction through content-based embedding.

MeLON performs superiorly on some recall metrics and worse on NDCG metrics. As a scheme that can adjust the learning rate in both directions on interactions and parameters to accommodate necessary samples, MeLON can better find the direction of gradient updates in less cold-start data and thus performs better in the MovieLens dataset. However, as the number of cold-start interactions increases in other datasets, its inability to find the commonality of cold-start items may bring inefficiency in updating.

5.2.2 Comparison of Popular Items. Meanwhile, we compare the metrics of PAM on popular items, the results are showed in Table 3.

It can be seen that the performance of the other baseline methods on popular items has improved compared to the cold-start items, reflecting the shift of the parameters toward the popular items due to more feedback on popular items. Meanwhile, PAM does not perform as well on popular as on cold-start items, demonstrating that the preference of parameter settings for cold-start tasks successfully optimizes on cold-start items. Notably, PAM still outperforms some baselines on popular items, demonstrating the superior performance of popular task parameters detached from the cold-start data for recommending items in the corresponding task.

Table 3: The reported top-K evaluation metric results on popular items on MovieLens dataset.

	R@5	R@10	R@20	N@5	N@10	N@20
PF	0.2949	0.3532	0.4285	0.2474	0.2661	0.2850
s^2Meta	0.2962	0.3544	0.4287	0.2488	0.2674	0.2861
IncCTR	0.2932	0.3515	0.4247	0.2466	0.2653	0.2838
SML	0.2729	0.3335	0.4049	0.2263	0.2459	0.2643
ASMG	0.2659	0.3249	0.3999	0.2210	0.2399	0.2588
MeLON	0.3205	0.4261	0.5139	0.2442	0.2689	0.3173
IMSR	0.2914	0.3581	0.4388	0.2317	0.2566	0.2821
PAM	0.3340	0.4031	0.4840	0.2771	0.2994	0.3198

5.3 Ablation Study (RQ2)

We compare PAM with several variants: **PAM-M** refers to the method without the cold-start task enhancer, **PAM-S** and **PAM-A** denote the method that adds only the self-supervision and data augmentation modules, respectively, and **PAM-F** is the method that implements the complete cold-start task enhancer.

We can see that both the optimization method for cold-start task information extraction with self-supervised signals and the data augmentation method for simulated cold-start data significantly enhance the PAM on all datasets. This reflects the nature of PAM's excellent task segmentation, which makes further optimization on a single task feasible.

For PAM-S, it utilizes self-supervised signals using the ID embedding information of popular items to guide the cold-start task. Compared with PAM-M, the improvement on the MovieLens dataset is relatively minor, while it has a significant optimization effect on the Yelp dataset. This may be because the high concentration of content-based features in the MovieLens dataset leads to the cold-start task itself being able to extract more information. In contrast, the decentralized content-based features in the Yelp dataset make the guidance approach more effective.

The PAM-A approach leverages the idea of data augmentation. Simulating cold-start consumption data using historical behavior-based embedding of popular items delivers a significant optimization compared to PAM-M across all datasets. It also shows that the sparseness of feedback on cold-start items is a significant and essential reason for the poor recommendation of cold-start items. Thus, the increase in data volume further enhances the task segmentation to improve the performance of cold-start items.

PAM-F combines all the proposed optimization methods for the cold-start task and ultimately brings significant improvements in the cold-start item metrics compared to the baseline. It is worth noting that the improvement effect of PAM-F is weakened compared with the combination of the respective improvement of PAM-S and PAM-A, which only incorporate a single loss. This is because both PAM-S and PAM-A optimize the cold-start task, but they use different methods and thus optimize in slightly different directions, thus weakening the effect of PAM-F's enhancement.

5.4 Breakdown Analysis (RQ3)

In this section, we conduct a breakdown analysis of PAM's ability to personalize parameters on tasks with different item popularity. To validate this, we summarize the following experiments. First, we saved the network parameters of cold-start and popular tasks generated by the corresponding task data. We sampled 200 batches and computed the top representations z_i in Sec 4.1.2 of cold-start and popular items using the corresponding parameters. Afterward, we categorized the input embedding into two types like in Eq. (13): behavior-based embedding e_q^{beh} and content-based embedding e_q^{con}. We mask the behavior-based and content-based embedding of the cold-started and popular items to 0, respectively, and compute the top representations again as inputs using the corresponding network parameters. The masked top representations are further compared with unmasked top representations. Fig. 3 illustrates the squared error of the top representations of the cold-start and popular items before and after masking the two types of embeddings.

In the task of cold-start items, the differences in the top representations of the network outputs before and after masking the behavior-based embedding were minor. In contrast, some dimensions of the top representations changed considerably before and after masking the content-based embedding. In cold-start items, content-based embedding stores less information due to the few-shot interactions in cold-start items. Hence, the parameters generated by the cold-start task can effectively extract the content information, which gives more importance to content-based rather than behavior-based embedding and improves the recommendation effect of the cold-start task.

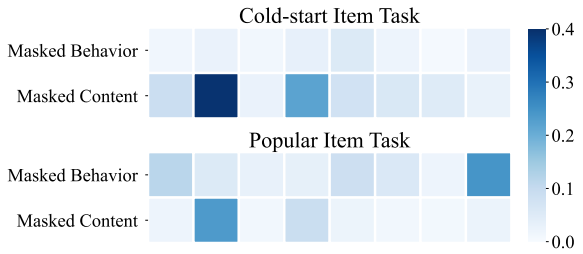


Figure 3: Squared errors of top representations of cold-start items and popular items before and after masking for different types of embedding inputs.

Table 4: Online performance of PAM compared to PF.

	Show%	LTR	CMTR	CLTR
PAM	+41.39%	+60.45%	+4.26%	+6.34%

In contrast, the top representations of the popular item tasks changed considerably before and after both embedding masks, and masking the behavior-based embedding introduced more significant error. This suggests that the network parameters of the popular item task can effectively utilize the information in both embedding and focus more on behavior-based embedding (*e.g.*, ID embedding).

We can also that masking the content-based embedding, both on the cold-start task and on the hot task, affects the exact same dimensions of the top-level representation. illustrating the similar extraction methods of content information of the cold-start and popular item tasks.

5.5 Online A/B Tests (RQ4)

To evaluate the performance of PAM in an online system, we deployed the PAM in a commercial, billion-user-scale, online recommender system and compared it to the PF baseline. In the online system, we counted the following metrics, including the rate of items appearing in users' recommendations (Show%), and the ratio of users liking (LTR), commenting (CMTR), and collecting videos (CLTR). The results of the experiments in Table 4 show that the high accuracy of PAM for item recommendation enables items to be better recommended to users who favor the item.

6 CONCLUSION

In this work, we find the problem of poor results of online recommender systems on cold-start items due to the long-tailed distribution of item popularity, as well as the inapplicability of existing cold-start schemes in scenarios with streaming data. To this end, we propose a unique PAM by partitioning the data into tasks by item popularity and optimizing the performance of cold-start task parameters while sharing information between tasks. Furthermore, we design a novel cold-start task enhancer to optimize the performance of cold-start tasks further by leveraging the popular item information. The conducted experiments demonstrate the superiority of the PAM approach, and in the future, we will further investigate different schemes to optimize the online cold-start problem.

REFERENCES

[1] Abien Fred Agfarap. 2019. Deep Learning using Rectified Linear Units (ReLU). arXiv:1803.08375 [cs.NE]

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2019. How to train your MAML. arXiv:1810.09502 [cs.LG]

[3] Homanga Bharadhwaj. 2019. Meta-Learning for User Cold-Start Recommendation. In 2019 International Joint Conference on Neural Networks (IJCNN). 1–8. https://doi.org/10.1109/IJCNN.2019.8852100

[4] Robin Devooght, Nicolas Kourtellis, and Amin Mantrach. 2015. Dynamic Matrix Factorization with Priors on Unknown Values. arXiv:1507.06452 [stat.ML]

[5] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Virtual Event, CA, USA) (KDD '20). Association for Computing Machinery, New York, NY, USA, 688–697. https://doi.org/10.1145/3394486.3403113

[6] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Anchorage, AK, USA) (KDD '19). Association for Computing Machinery, New York, NY, USA, 2895–2904. https://doi.org/10.1145/3292500.3330726

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. arXiv:1703.03400 [cs.LG]

[8] Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. 2022. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 3953–3957.

[9] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interact. Intell. Syst. 5, 4, Article 19 (dec 2015), 19 pages. https://doi.org/10.1145/2827872

[10] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging Language and Items for Retrieval and Recommendation. arXiv preprint arXiv:2403.03952 (2024).

[11] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2333–2338.

[12] SeongKu Kang, Junyoung Hwang, Dongha Lee, and Hwanjo Yu. 2019. Semi-Supervised Learning for Cross-Domain Recommendation to Cold-Start Users. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (Beijing, China) (CIKM '19). Association for Computing Machinery, New York, NY, USA, 1563–1572. https://doi.org/10.1145/3357384.3357914

[13] Minseok Kim, Hwanjun Song, Yooju Shin, Dongmin Park, Kijung Shin, and Jae-Gil Lee. 2022. Meta-Learning for Online Update of Recommender Systems. arXiv:2203.10354 [cs.IR]

[14] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/ARXIV.1412.6980

[15] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Anchorage, AK, USA) (KDD '19). Association for Computing Machinery, New York, NY, USA, 1073–1082. https://doi.org/10.1145/3292500.3330859

[16] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on Heterogeneous Information Networks for Cold-start Recommendation. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Virtual Event, CA, USA) (KDD '20). Association for Computing Machinery, New York, NY, USA, 1563–1573. https://doi.org/10.1145/3394486.3403207

[17] Haokai Ma, Zhuang Qi, Xinxin Dong, Xiangxian Li, Yuze Zheng, and Xiangxu Mengand Lei Meng. 2023. Cross-Modal Content Inference and Feature Enrichment for Cold-Start Recommendation. arXiv:2307.02761 [cs.IR]

[18] Mohammmadmahdi Maheri, Reza Abdollahzadeh, Bardia Mohammadi, Mina Rafiei, Jafar Habibi, and Hamid R. Rabiee. 2023. ClusterSeq: Enhancing Sequential Recommender Systems with Clustering based Meta-Learning. arXiv:2307.13766 [cs.IR]

[19] Robert K. Merton. 1968. The Matthew Effect in Science. Science 159, 3810 (1968), 56–63. https://doi.org/10.1126/science.159.3810.56 arXiv:https://www.science.org/doi/pdf/10.1126/science.159.3810.56

[20] Krishna Prasad Neupane, Ervine Zheng, Yu Kong, and Qi Yu. 2022. A Dynamic Meta-Learning Model for Time-Sensitive Cold-Start Recommendations. arXiv:2204.00970 [cs.IR]

[21] Xingyu Pan, Yushuo Chen, Changxin Tian, Zihan Lin, Jinpeng Wang, He Hu, and Wayne Xin Zhao. 2022. Multimodal Meta-Learning for Cold-Start Sequential Recommendation. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management (Atlanta, GA, USA) (CIKM '22). Association for Computing Machinery, New York, NY, USA, 3421–3430. https://doi.org/10.1145/3511808.3557101

[22] Deepak Kumar Panda and Sanjog Ray. 2022. Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review. J. Intell. Inf. Syst. 59, 2 (oct 2022), 341–366. https://doi.org/10.1007/s10844-022-00698-5

[23] Haoyu Pang, Fausto Giunchiglia, Ximing Li, Renchu Guan, and Xiaoyue Feng. 2022. PNMTA: A Pretrained Network Modulation and Task Adaptation Approach for User Cold-Start Recommendation. In Proceedings of the ACM Web Conference 2022 (Virtual Event, Lyon, France) (WWW '22). Association for Computing Machinery, New York, NY, USA, 348–359. https://doi.org/10.1145/3485447.3511963

[24] Danni Peng, Sinno Jialin Pan, Jie Zhang, and Anxiang Zeng. 2021. Learning an Adaptive Meta Model-Generator for Incrementally Updating Recommender Systems. In Proceedings of the 15th ACM Conference on Recommender Systems (Amsterdam, Netherlands) (RecSys '21). Association for Computing Machinery, New York, NY, USA, 411–421. https://doi.org/10.1145/3460231.3474239

[25] Shameem A Puthiya Parambath and Sanjay Chawla. 2020. Simple and effective neural-free soft-cluster embeddings for item cold-start recommendations. Data Mining and Knowledge Discovery 34 (09 2020). https://doi.org/10.1007/s10618-020-00708-6

[26] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. 2020. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. In International Conference on Learning Representations. https://openreview.net/forum?id=rkgMkCEtPB

[27] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. IEEE Trans. on Knowl. and Data Eng. 31, 2 (feb 2019), 357–370. https://doi.org/10.1109/TKDE.2018.2833443

[28] Xuehan Sun, Tianyao Shi, Xiaofeng Gao, Yanrong Kang, and Guihai Chen. 2021. FORM: Follow the Online Regularized Meta-Leader for Cold-Start Recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (Virtual Event, Canada) (SIGIR '21). Association for Computing Machinery, New York, NY, USA, 1177–1186. https://doi.org/10.1145/3404835.3462831

[29] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748 [cs.LG]

[30] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. In Proceedings of the 31st International Conference on Neural Information Processing Systems (Long Beach, California, USA) (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6907–6917.

[31] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In Proceedings of the 31st International Conference on Neural Information Processing Systems (Long Beach, California, USA) (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 4964–4973.

[32] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2020. A Practical Incremental Method to Train Deep CTR Models. arXiv:2009.02147 [cs.IR]

[33] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Tie-Yan Liu, and Wei Chen. 2013. A Theoretical Analysis of NDCG Type Ranking Measures. arXiv:1304.6480 [cs.LG]

[34] Zhikai Wang and Yanyan Shen. 2023. Incremental Learning for Multi-Interest Sequential Recommendation. In 2023 IEEE 39th International Conference on Data Engineering (ICDE). 1071–1083. https://doi.org/10.1109/ICDE55515.2023.00087

[35] Zhenchao Wu and Xiao Zhou. 2023. M2EU: Meta Learning for Cold-start Recommendation via Enhancing User Preference Estimation. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23). Association for Computing Machinery, New York, NY, USA, 1158–1167. https://doi.org/10.1145/3539618.3591719

[36] Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. 2022. FIRE: Fast Incremental Recommendation with Graph Signal Processing. In Proceedings of the ACM Web Conference 2022 (Virtual Event, Lyon, France) (WWW '22). Association for Computing Machinery, New York, NY, USA, 2360–2369. https://doi.org/10.1145/3485447.3512108

[37] Ruobing Xie, Yalong Wang, Rui Wang, Yuanfu Lu, Yuanhang Zou, Feng Xia, and Leyu Lin. 2022. Long Short-Term Temporal Meta-learning in Online Recommendation. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (Virtual Event, AZ, USA) (WSDM '22). Association for Computing Machinery, New York, NY, USA, 1168–1176. https://doi.org/10.1145/3488560.3498371

[38] Chen Yang, Jin Chen, Qian Yu, Xiangdong Wu, Kui Ma, Zihao Zhao, Zhiwei Fang, Wenlong Chen, Chaosheng Fan, Jie He, Changping Peng, Zhangang Lin, and Jingping Shao. 2023. An Incremental Update Framework for Online Recommenders with Data-Driven Prior. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (Birmingham, United Kingdom) (CIKM '23). Association for Computing Machinery, New York, NY,

USA, 4894–4900. https://doi.org/10.1145/3583780.3615456

[39] Junliang Yu, Min Gao, Jundong Li, Hongzhi Yin, and Huan Liu. 2018. Adaptive Implicit Friends Identification over Heterogeneous Network for Social Recommendation. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (Torino, Italy) (CIKM '18). Association for Computing Machinery, New York, NY, USA, 357–366. https://doi.org/10.1145/3269206.3271725

[40] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to Retrain Recommender System? A Sequential Meta-Learning Method. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (Virtual Event, China) (SIGIR '20). Association for Computing Machinery, New York, NY, USA, 1479–1488. https://doi.org/10.1145/3397271.3401167

[41] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. 2020. Cold-start Sequential Recommendation via Meta Learner. arXiv:2012.05462 [cs.IR]

[42] Yongchun Zhu, Ruobing Xie, Fuzhen Zhuang, Kaikai Ge, Ying Sun, Xu Zhang, Leyu Lin, and Juan Cao. 2021. Learning to Warm Up Cold Item Embeddings for Cold-start Recommendation with Meta Scaling and Shifting Networks. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (Virtual Event, Canada) (SIGIR '21). Association for Computing Machinery, New York, NY, USA, 1167–1176. https://doi.org/10.1145/3404835.3462843

A APPENDIX

A.1 Experimental Settings

A.1.1 Datasets. We selected three public datasets that containing timestamp information.

MovieLens [9] is a dataset of user ratings for movies. We selected one of the MovieLens 25M datasets and retained the data for the period 2014-2018 for training, containing 6,840,091 interactions between 43,181 users and 51,142 movies.

Yelp is a dataset containing user reviews of various businesses spanning over 10 years, containing 6,990,280 interactions between 1,987,929 users and 150,346 businesses.

Book [10] is a part of the **Amazon** dataset, which collects book reviews from purchasers on the Amazon e-shopping site. We selected data from the period 2012-2023 for training and retained users with 10 or more interactions and items with 3 and more interactions, remaining 6,402,728 interactions between 351,487 users and 581,717 books.

A.1.2 Baselines. We compare our proposed PAM with the following baseline, where all scenarios leverage the previously mentioned two-tower model structure and the feature inputs remain the same.

- **Periodical Fine-tuning (PF)** or incremental updating, is a frequently used type of update in online systems, which performs an update on recommender system using the incoming data periodically to serve the next moment.

- **s^{2}Meta [6]** employs a novel meta-learning task division approach that considers item scenarios as tasks to achieve better recommendation results for new scenarios.

- **IncCTR [32]** is a practical incremental method that leverages knowledge distillation for training, use the current knowledge stored in model to guide the retraining.

- **SML [40]** proposes a meta-model that takes the previous moment's recommender and current moment's data as inputs and generate the next moment's recommender for serving through a CNN-based network.

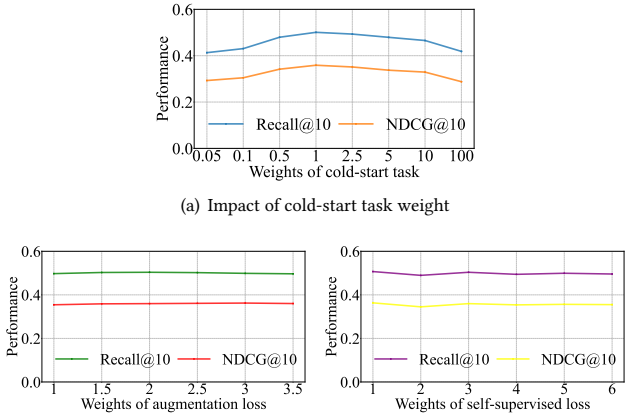
- **ASMG [24]** better captures long-term interests through historical recommenders to generate models for serving through a meta-model generator that introduces GRU.

- **MeLON [13]** generates more stable online recommendation models by maintaining dynamic learning rates for different parameters and distinguishing the importance of interaction data.

- **IMSR [34]** allows the traditional adaptive interest capture model to better capture new interests of users periodically by introducing unique interest shifters.

A.2 Parameters Sensitivity Study (RQ5)

We analyzed the parameter sensitivity of PAM in two parameter dimensions: the weight of the cold-start task λ and the weight between the losses of different modules γ, the results are displayed in Fig. 4.



(a) Impact of cold-start task weight

(b) Impact of augmentation loss weight (c) Impact of self-supervised loss weight

Figure 4: Performance of PAM *w.r.t.* different weights of cold-start tasks and different weights of losses.

A.2.1 Impact of cold-start task weight. We evaluated the impact of different weights for the cold-start task in calculating the loss of meta-learning in Fig. 4(a). It can be seen that when the weight of the cold-start task is set in an appropriate range, it can effectively improve the performance on cold-start items. If the weight of the cold-start task is set too low, the loss on the cold-start parameter will take a minor percentage of the overall loss during the global update, thus making the parameter less specialized for the cold-start task. On the other hand, if the weights of the cold-start tasks are set too high, then as mentioned before, most of the information in the popular data will be lost, leaving the parameters shared between the tasks poorly updated, leading to the same inferior performance.

A.2.2 Impact of modules loss weight. We similarly analyzed the effect of the loss weights of different modules on the effectiveness of PAM. It can be seen that for the data-enhanced modules in Fig. 4(b), the enhancement brought are more stable and the weights only have minor effect on the metric performance. Besides, the self-supervised guidance module, although performance stably, fluctuates slightly more than the data enhancement module, as shown in Fig. 4(c).