

Deployable and Continuable Meta-learning-Based Recommender System with Fast User-Incremental Updates

Renchu Guan
College of Computer Science and Technology, Jilin University
Changchun, China
guanrenchu@jlu.edu.cn

Haoyu Pang
College of Computer Science and Technology, Jilin University
Changchun, China
panghy20@mails.jlu.edu.cn

Fausto Giunchiglia
DISI, University of Trento
Trento, Italy
fausto.giunchiglia@unitn.it

Ximing Li
College of Computer Science and Technology, Jilin University
Changchun, China
ximingli@jlu.edu.cn

Xuefeng Yang
Tencent
Shenzhen, China
ryanxfyang@tencent.com

Xiaoyue Feng*
College of Computer Science and Technology, Jilin University
Changchun, China
fengxy@jlu.edu.cn

ABSTRACT

User cold-start is a major challenge in building personalized recommender systems. Due to the lack of sufficient interactions, it is difficult to effectively model new users. One of the main solutions is to obtain an initial model through meta-learning (mainly gradient-based methods) and adapt it to new users with a few steps of gradient descent. Although these methods have achieved remarkable performance, they are still far from being usable in real-world applications due to their high-demand data processing, heavy computational burden, and inability to perform effective user-incremental update. In this paper, we propose a **d**eployable and **c**ontinuable **me**ta-learning-based **r**ecommendation (DCMR) approach, which can achieve fast user-incremental updating with task replay and first-order gradient descent. Specifically, we introduce a dual-constrained task sampler, distillation-based loss functions, and an adaptive controller in this framework to balance the trade-off between stability and plasticity in updating. In summary, DCMR can be updated while serving new users; in other words, it learns continuously and rapidly from a sequential user stream and is able to make recommendations at any time. The extensive experiments conducted on three benchmark datasets illustrate the superiority of our model.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; Data mining;
• **Computing methodologies** → *Machine learning*.

KEYWORDS

Recommender system; Cold-start problem; Meta-learning; Incremental training

ACM Reference Format:
Renchu Guan, Haoyu Pang, Fausto Giunchiglia, Ximing Li, Xuefeng Yang, and Xiaoyue Feng. 2022. Deployable and Continuable Meta-learning-Based Recommender System with Fast User-Incremental Updates. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531964

1 INTRODUCTION

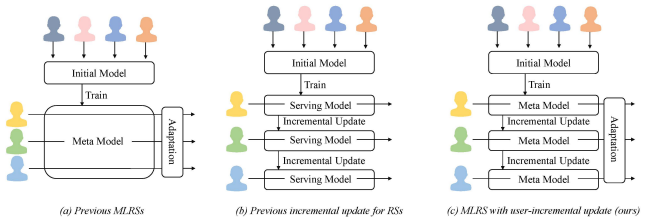


Figure 1: Illustration of the MLRS with user-incremental update.

Recommender systems (RSs) [5, 14, 39] have been applied widely for various online services, such as e-commerce [12, 47] and news portals [1, 23], to address the issue of information overload. They make recommendations for users by estimating their preferences from historical interactions. However, when new users appear, the RSs cannot accurately model them due to the lack of sufficient interactions, which is the so-called user cold-start problem [32, 35, 48]. Meta-learning [11, 31, 33], especially gradient-based meta-learning [10, 22], is an effective solution for this problem. Meta-learning-based RSs (MLRSs)[1] [2, 16, 20, 24] typically learn a common initialization for all users and generate task-specific models for each user with a few steps of adaptation.

*Corresponding author.

[1]The MLRS covers a wide range of methods, but here, it specifically refers to gradient-based methods to avoid redundant narratives (as described in Section 5).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '22, July 11–15, 2022, Madrid, Spain
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531964

Most existing MLRSs are designed in the context of an offline setting, which means that new users who arrive during serving will not be adopted to continuously optimize the meta-model. However, in the real world, new users always appear in streams instead of at the same time [25, 44]. Therefore, a practical RS should be able to serve or update at any time, that is, should be able to continuously perform user-incremental updating from the user stream (Figure 1). Additionally, most MLRSs rely on estimating the second derivatives via support and query sets [7, 16, 36, 43], which leads to high-demand data processing and a heavy computational burden during update. This pushes MLRSs farther from mainstream industrial deployment. Considering the abovementioned factors, we require the following four properties for a user-incremental MLRS to qualify as a deployable framework:

1) *Ability to prevent catastrophic forgetting.* A major challenge for incremental learning is the stability-plasticity dilemma [17]. RSs should make the best use of new users for updating without forgetting the knowledge learned from old users [15, 28, 41].
2) *Stable and fast user-incremental updating.* The updating should be almost real-time [6] so that the method can absorb new users and follow social trends as quickly as possible, and the time consumption of each update step should not increase with the number of steps [3, 25, 44].
3) *Limited and preset memory consumption.* We allow the model to occupy a small amount of additional memory, but this memory should not continue to grow over time, and its size should be able to be set at the beginning [30].
4) *Ease of implementation, ease of computation, and model agnosticism.* The meta-optimization framework should be extremely versatile and should not impose too many restrictions on the data, models, and computing environment.

Currently, three types of incremental RSs are widely adopted, which we transfer to MLRSs: **full retraining** (FR), **sample-based retraining** (SR), and **fine-tuning** (FT) RSs. FR (retraining the model based on all data, including new and old users) is the theoretical upper bound of user-incremental updating without considering the sequence. However, it clearly violates requirements 2) and 3). FT (updating the RSs based on new users only [29, 37]) does meet requirements 2) and 3), but it violates 1). SR (updating the RSs based on new users and sampled old users) is a compromise between FR and FT, but it does not usually integrate the advantages of the two methods well [4, 38]. Most of the existing gradient-based MLRSs do not meet 4).

In this work, we propose a novel **d**eployable and **c**ontinuable **MLR**S (DCMR) framework to obtain user-incremental MLRSs that meet all criteria above. Specifically, DCMR builds task memory with a limited size through a dual-constrained task sampler and replays these tasks during user-incremental updating. Additionally, DCMR is a model-agnostic first-order MLRS framework without support and query sets, which is fast in computation, simple in implementation and extremely flexible. Of course, establishing such a framework is nontrivial, presenting us with three key challenges:

i. *How can the model be prevented from overfitting the replay tasks?* The replay tasks are sampled from the overall tasks [45]. With user-incremental updating, the model will overfit the remaining tasks and forget the knowledge obtained from other old tasks [19].
ii. *How can the model be made to yield different acceptance levels for new and replay tasks?* In an ideal case, the model should treat these two types of tasks differently: it should actively absorb knowledge from the new tasks and only be conservative about the replay tasks to preserve previous knowledge.
iii. *How can the stability and plasticity of the model be adaptively adjusted?* As the update progresses, the model should be more plastic in the early stage, while during the later stage, it must avoid any drastic changes since the generic knowledge for most users will have been learned [27].

In DCMR, for i, the adaptive models of replay tasks are updated locally by minimizing the distillation loss that, for replay tasks, encourages the reproduction of the implicit knowledge stored in the meta model. To achieve this, we specifically design two distillation losses for the feedback of RSs. For ii, in the local update, the representation parameters are updated for all tasks to promote a shared generic embedding space among users, but the predictor parameters are updated only for new tasks to avoid interference from replay tasks. For iii, the change to the model due to the new user is dynamically managed by a momentum-based adaptive controller (similar to a dynamic learning rate).

Finally, the explanation from an optimal solution manifold perspective and the analysis of time complexity prove the validity and deployability of DCMR. Extensive experiments on three benchmark datasets show that DCMR significantly outperforms the existing state-of-the-art (SOTA) RSs.

2 PRELIMINARIES

In this section, we formally define user-incremental updating in MLRSs and introduce the network structure of the base model.

2.1 Problem Definition

In real-world RSs, new users (cold-start users) continue to enter the systems over time. The previous MLRSs usually generate a task-specific model for each new user based on the knowledge learned from old users but cannot continuously learn via the new users. To maximize data utilization and maintain the real-time performance of the model, a common method is to periodically perform user-incremental updating on MLRSs.

In the context of MLRSs, we define the task τ as *making recommendations for one user* [16]. Formally, τ_k includes a user u_k and his or her interaction set $\mathcal{X}_k = \{(u_k, i_n, y_{k,n})\}_{n=1}^{N}$ with the corresponding item set $\mathcal{I}_k = \{i_{k,n}\}_{n=1}^{N}$ and label set $\mathcal{Y}_k = \{y_{k,n}\}_{n=1}^{N}$, where N is the size of $\mathcal{X}_k$. In serving, the trained meta model learns a set of personalized parameters for each new user through task adaptation. For streaming scenes, the user stream is represented as $\{U_1, U_2, ..., U_t, U_{t+1}, ...\}$, where U_t denotes the users and their interactions newly collected by the system in time period t or the t-th batch of users in the time series, which depends on whether the system is updated with a fixed time period or with a fixed number of new users [25]. Let the meta model parameters before the t-th update be Φ_{t-1}. The MLRS updates its parameters to Φ_t via the new user set U_t and the previous users $\{U_1, ..., U_{t-1}\}$ and serves the new

users in set U_{t+1}. This process is formalized as:

$$(\Phi_{t-1}, U_t, \{U_1, ..., U_{t-1}\}) \rightarrow update \rightarrow \Phi_t \rightarrow serve \rightarrow U_{t+1} \quad (1)$$

Clearly, this is a reckless FR paradigm. We confirm that the meta model with parameters Φ_{t-1} has extracted useful "knowledge" from the previous users, so it is unwise to reuse the old data for updating. A more elegant approach is to sample a small number of representative users from the old data to form a replay task memory $\mathcal{M}_{t-1} = \{u_m\}_{m=1}^{M}$, where M is the size of this memory. During the update, the model can reproduce the distribution of the old data by recalling these reply tasks. In addition, the model can implicitly learn the order in which users appear. Thus, we reformulate the user-incremental updating as:

$$(\Phi_{t-1}, U_t, \mathcal{M}_{t-1}) \rightarrow update \rightarrow \Phi_t \rightarrow serve \rightarrow U_{t+1}. \quad (2)$$

It is worth noting that once the model completes one update step and the task memory is reconstructed, the unreserved data and the previous model will no longer be visible.

2.2 Model Structure

In principle, DCMR is largely model agnostic and compatible with most recommender networks. However, here, we provide a specific definition of the base model to avoid overly general notation.

The base model $\mathcal{F}_\Phi$ (with parameters Φ) consists of two components: a **representation** module $\mathcal{R}_\theta$ (with parameters θ) and a **prediction** module $\mathcal{P}_\omega$ (with parameters ω) (Figure 2a). DCMR uses the representation module to generate embedding vectors for users and items, and the prediction module performs feature crosses to predict the interaction between users and items. This process is expressed as $\hat{y} = \mathcal{F}_\Phi(u, i) = \mathcal{P}_\omega(\mathcal{R}_\theta(u, i))$,where $\Phi = \{\omega; \theta\}$.

Representation $\mathcal{R}_\theta$: First, we define two representation modules for different scenarios. For scenes with side information of users and items (such as the user's occupation or the genre of a movie), we use the corresponding weight matrix E to convert each feature c into a feature vector. The representation module is defined as:

$$\mathcal{R}_\theta(u, i) = \{E_1^u c_1^u; ...; E_p^u c_p^u; E_1^i c_1^i; ...; E_q^i c_q^i\}, \quad (3)$$

where p/q is the number of features of each user/item and $\{;\}$ is the connection operation. When only the IDs of users and items are available, we redefine the representation module as:

$$\mathcal{R}_\theta(u, i) = \{\sigma(W_2^u \sigma(W_1^u c_u + b_1^u) + b_2^u); \sigma(W_2^i \sigma(W_1^i c_i + b_1^i) + b_2^i)\} \quad (4)$$

where W and b represent the weight matrices and bias vectors, respectively, σ is the sigmoid activation function, and $c_{u/i}$ is the one-hot ID of user u/item i.

Prediction $\mathcal{P}_\omega$: Then, we give the corresponding prediction modules for the two feedback modes of the RSs. For explicit feedback scenarios (such as ratings), we define $\mathcal{P}_\omega$ as:

$$\hat{y} = \mathcal{P}_\omega(v) = MLP(v), \quad (5)$$

where $\hat{y}$ is the prediction score given by the model, MLP is a multilayer perceptron, and v is the embedding vector from $\mathcal{R}_\theta$. Based on this, we add an activation function for implicit feedback scenarios (such as clicking):

$$\hat{y} = \mathcal{P}_\omega(v) = \sigma MLP(v). \quad (6)$$

The output $\hat{y}$ is a value in $(0, 1)$, which represents the predicted interaction probability.

3 DCMR

In this section, we describe the procedure of DCMR (Algorithm 1) and provide an explanation of its validity. First, Section 3.1 introduces the details of user-incremental updating. Then, Section 3.2 provides a method of constructing the replay task memory. Next, Section 3.3 describes the initial training and online serving of DCMR. Finally, Section 3.4 analyzes the principle of DCMR to achieve user-incremental updating.

Algorithm 1: DCMR Framework.

Input: Initial user set U_0, user stream $\{U_1, U_2, ..., U_t, ...\}$.
Output: meta models $\{\Phi_1, \Phi_2, ..., \Phi_t, ...\}$.

```
Φ_0 ← INITIAL TRAINING
M_0 ← CONSTRUCT MEMORY
for U_t in user stream do
    Online:
    U_t ← SERVING
    Offline:
    Φ_t ← USER-INCREMENTAL UPDATE (Algorithm 2)
    M_t ← RECONSTRUCT MEMORY (Algorithm 3)
end
```

3.1 User-Incremental Updating

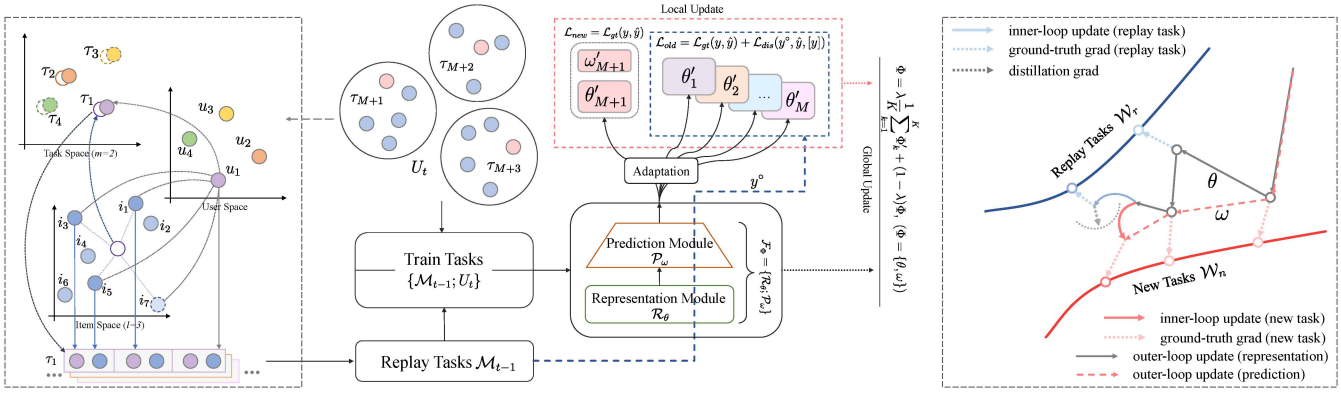
DCMR involves two updates (Figure 2b and Algorithm 2), a **local** update that generates task-specific models for each user from the meta model through a few steps of gradient-based adaptation and a **global** update that fuses task-specific models into the meta model.

Local update[2]: In the local update of DCMR, we randomly sample a batch of users from the new user set U_t and the replay task memory $\mathcal{M}_{t-1}$ to construct an augmented task set $\mathcal{T} = \{\tau_k\}_{k=1}^{K} = \{(u_k, \mathcal{I}_k, \mathcal{Y}_k)\}_{k=1}^{K}$, which contains the replay tasks $\{\tau_1, ..., \tau_M\}$ and the new tasks $\{\tau_{M+1}, ...\}$. Then, each task τ_k is used to generate corresponding task-specific parameters $\Phi_k' = \{\theta_k', \omega_k'\}$ from the meta model parameters $\Phi_o = \{\theta_o, \omega_o\}$ by minimizing a loss function through stochastic gradient descent (SGD).[3] Here, θ is updated for all tasks, but ω is updated only for new tasks. For each new task, the loss function is set to the mean square error (MSE) or binary cross-entropy (BCE) according to the feedback type to encourage the model to output correct results. The loss function $\mathcal{L}_{new}$ for new tasks is defined as:

$$\mathcal{L}_{new}(\tau_k) = \begin{cases} \mathcal{L}_{MSE}(\tau_k), & for\ explicit\ feedback; \\ \mathcal{L}_{BCE}(\tau_k), & for\ implicit\ feedback \end{cases} \quad (7)$$

$$\mathcal{L}_{MSE}(\tau_k) = \frac{1}{N} \sum_{n=1}^{N} \left\| y_{k,n} - \hat{y}_{k,n} \right\|_2^2 \quad (8)$$

[2]DCMR performs a multistep local update, but we show the one-step version here for simplicity.
[3]That is, Φ_{t-1} in Section 2.1. To avoid confusing references, we use Φ_o here to emphasize that it is the **original** parameter in the t-th user-incremental updating step.



(a) The construction of replay task memory *(b) The user-incremental update of DCMR* *(c) The illustration of DCMR update*

Figure 2: The overall framework of DCMR and its explanation from an optimal solution manifold perspective.

$$\mathcal{L}_{BCE}(\tau_k) = \frac{1}{N}\sum_{n=1}^{N}[y_{k,n}log(\hat{y}_{k,n}) + (1-y_{k,n})log(1-\hat{y}_{k,n})] \quad (9)$$

$$\hat{y}_{u,k} = \mathcal{F}_{\Phi'}(u_k, i_{k,n}) \quad (10)$$

where $y_{k,n}$ and $\hat{y}_{k,n}$ are the ground truth and predicted label of the interaction. Then, the local parameters Φ'_k for the new user u_k are generated by:

$$\Phi'_k = \Phi_o - \alpha\nabla_{\Phi_o}\mathcal{L}_{new}(\tau_k), \quad (11)$$

where α denotes the learning rate of local update. For replay tasks, we adopt an additional distillation loss to obtain task-specific models closer to a general task manifold for all old users while encouraging the correct output. We customize two novel distillation losses for different feedback types of RSs. The loss function $\mathcal{L}_{re}$ for replay tasks is defined as follows:

$$\mathcal{L}_{re}(\tau_k) = \begin{cases} \mathcal{L}_{MSE}(\tau_k) + \beta\mathcal{L}_{exp-dis}(\tau_k), & for\ explicit\ feedback; \\ \mathcal{L}_{BCE}(\tau_k) + \beta\mathcal{L}_{imp-dis}(\tau_k), & for\ implicit\ feedback \end{cases} \quad (12)$$

where $\mathcal{L}_{exp-dis}$ and $\mathcal{L}_{imp-dis}$ are the distillation losses for explicit and implicit feedback, respectively. Let $y^{\circ}_{k,n}$ denote the output of the meta model, where ϵ is a margin. We set $\mathcal{L}_{exp-dis}(\tau_k)$ as a bounded promotion function, which promotes local updates only when the task-specific loss is larger than that of the meta model:

$$\mathcal{L}_{exp-dis}(\tau_k) = \frac{1}{N}\sum_{n=1}^{N} \begin{cases} \left\|y^{\circ}_{k,n} - \hat{y}_{k,n}\right\|_2^2, & if\ \left\|y^{\circ}_{k,n} - \hat{y}_{k,n}\right\|_2^2 + \epsilon > \left\|y^{\circ}_{k,n} - \hat{y}_{k,n}\right\|_2^2 \\ 0, & otherwise \end{cases} \quad (13)$$

Similar to traditional knowledge distillation methods, $\mathcal{L}_{imp-dis}(\tau_k)$ is set to encourage the model to reproduce the logit output of the meta model:

$$\mathcal{L}_{imp-dis}(\tau_k) = \frac{1}{N}\sum_{n=1}^{N}[y^{\circ}_{k,n}log(\hat{y}_{k,n}) + (1-y^{\circ}_{k,n})log(1-\hat{y}_{k,n})] \quad (14)$$

$$y^{\circ}_{k,n} = \mathcal{F}_{\Phi_o}(u_k, i_{k,n}) \quad (15)$$

One might object to this distillation-based method on the grounds that it is equivalent to directly reducing the learning rate α. However, note that reducing α preserves only the meta-model parameters, but the role of distillation loss is to activate the *knowledge* contained in the meta model. Then, the local update of the replay tasks can be expressed as:

$$\omega'_k = \omega_o \quad (16)$$

$$\theta'_k = \theta_o - \alpha\nabla_{\theta_o}\mathcal{L}_{re}(\tau_k) \quad (17)$$

Global update: In the global update of DCMR, we combine the initial meta model and all task-specific models into a new meta model. Specifically, we treat the distance $(\Phi_o - \Phi'_k)$ as the gradient of global update and perform SGD on the meta model.

$$\Phi_o = \Phi_o - \lambda\frac{1}{K}\sum_{k=1}^{K}(\Phi_o - \Phi_k) = \lambda\frac{1}{K}\sum_{k=1}^{K}\Phi_k + (1-\lambda)\Phi_o \quad (18)$$

where λ denotes the learning rate of the global update, that is, the model fusion controller. Simply put, the purpose of global update is to find an average direction and move Φ_o to a new point that is close to all new and old tasks (as explained in Section 3.4). Moreover, the model should dynamically maintain the right balance between plasticity and stability according to the current state. Ideally, the model should actively accept new tasks in the early stages and tend to remain stable in the later stages. Therefore, we set λ as a momentum-based adaptive controller as follows:

$$\lambda = exp(-\eta\frac{t}{T}) \quad (19)$$

where T is the total number of update stages and η is a hyperparameter that controls the descending speed of λ. Since T is not always known, we do not set it in DCMR but in an additional model DCMR+.

3.2 Replay Task Memory

After each update step, we reconstruct the replay task memory based on a dual-constrained strategy (Figure 2a and Algorithm 3). Specifically, for user u_k, we first compute prototype ρ_k of item set

Algorithm 2: User-incremental updating of DCMR.

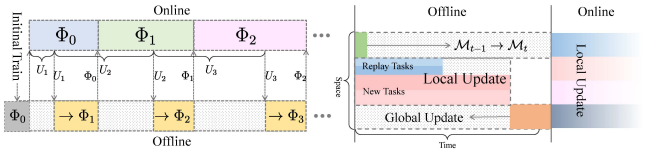
Input: Replay task memory $\mathcal{M}_{t-1}$, user set U_t, previous parameters Φ_{t-1}, hyperparameters α, λ, and η.
Output: Updated parameters Φ_t.

while *not converged* **do**
  for $\tau \in \{\tau \sim P(U_t)\} \cup \{\tau \sim P(\mathcal{M}_{t-1})\}$ **do**
    Calculate $\mathcal{Y}^\circ$ with Eq. 15;
    Predict $\hat{\mathcal{Y}}$ with Eq. 10;
    Local update:
    if τ *is a replay task* **then**
      Calculate loss $\mathcal{L}_{re}$ with Eq. 12;
      Update $\Phi \rightarrow \Phi'$ with Eq. 16 and 17;
    end
    if τ *is a new task* **then**
      Calculate loss $\mathcal{L}_{new}$ with Eq. 7;
      Update $\Phi \rightarrow \Phi'$ with Eq. 11;
    end
    Global update:
    Generate meta model parameters Φ_t with Eq. 18;
  end
end

Algorithm 3: Reconstructing the replay task memory.

Input: Replay task memory $\mathcal{M}_{t-1}$, new user set U_t, hyperparameters L and M.
Output: Replay task memory $\mathcal{M}_t$.

for $\tau \in \{U_t\}$ **do**
  Calculate the prototype ρ with Eq. 20;
end
Sample users from $\{\mathcal{M}_{t-1}, U_t\}$ to reconstruct $\mathcal{M}_t$ with Eq. 21;
Remove redundant items for each user in $\mathcal{M}_t$.



(a) The overall procedure of DCMR. *(b) One update step of DCMR.*

Figure 3: The online and offline settings of DCMR.

I_k.

$$\rho_k = \arg\min_{\rho} \sum_{n=1}^{N} \|\rho - i_n\| \tag{20}$$

where i_n is the item embeddings obtained via the representation module of the meta model. Next, we impose the first constraint on the user side by uniformly sampling M users from the distribution $P(\mathcal{M}_{t-1}, U_t)$. We impose the second constraint on the item side by selecting the top L items closest to the prototype ρ_k for each sampled user u_k.

Thus far, we have completed the reconstruction of the replay task memory, which is expressed as:

$$\mathcal{M}_t = \{\tau_m\}_{m=1}^{M} = \{(u_m, \mathcal{I}_m, \mathcal{Y}_m)\}_{m=1}^{M} \tag{21}$$

$$\mathcal{I}_m = \{i_{m,l}\}_{l=1}^{L} \tag{22}$$

$$\mathcal{Y}_m = \{y_{m,l}\}_{l=1}^{L} \tag{23}$$

Clearly, each user can be given a time-related weight to reflect the evolution of social preferences. However, the replay tasks in $\mathcal{M}_{t-1}$ are sampled from the old tasks, and the new tasks are directly added to the sampling pool together with the replay tasks. The whole sampling process can be expressed as:

$$\mathcal{M}_t \sim P(M_{t-1}, U_t) = P(P(\mathcal{M}_{t-2}, U_{t-1}), U_t). \tag{24}$$

Therefore, the old tasks have implicitly gained a lower sampling probability. On the principle that *"entities should not be multiplied unnecessarily" (Ockham's razor)*, we choose this more concise method.

3.3 Initial Training and Online Serving

In the initial training, DCMR treats all users as new users to train the initial meta model and samples from these users to construct the replay task memory. During online serving, DCMR generates the task-specific model for each new user through local update.

3.4 Explanation and Analysis

In this section, we provide an explanation of DCMR from an optimal solution manifold perspective, analyze its time complexity, and compare it with other meta-optimization algorithms.

The core idea of DCMR is to find a point that is close to all optimal solution manifolds of old and new tasks via iterative local and global updates (Figure 2c). Let $\mathcal{W}_k$ denote the optimal solution for τ_k. The optimization goal of DCMR in terms of Euclidean distance can be expressed as:

$$\min_{\Phi_o} \mathbb{E}_k[\frac{1}{2}\mathcal{D}(\Phi_o, \mathcal{W}_k)^2]. \tag{25}$$

Note that the distance $\mathcal{D}(\Phi_o, \mathcal{W}_k)$ is for a single point, which is different from the loss $\mathcal{L}_k(\mathcal{F}_{\Phi_o})$ between high-dimensional manifolds. Therefore, DCMR follows a different optimization goal than pretraining. Then, we can compute the gradient as:

$$\nabla_{\Phi_o} \mathbb{E}_k[\frac{1}{2}\mathcal{D}(\Phi_o, \mathcal{W}_k)^2] = \mathbb{E}_k[\Phi_o - \mathcal{W}_k]. \tag{26}$$

Next, we perform gradient descent on τ_k as follows:

$$\Phi_o = \Phi_o - \lambda(\Phi_o - \mathcal{W}_k) = (1-\lambda)\Phi_o + \lambda\mathcal{W}_k. \tag{27}$$

In fact, we cannot directly compute $\mathcal{W}_k$, so we replace it with its approximation, which is obtained by performing a few steps of gradient descent on Φ_o.

$$\mathcal{W}_k \leftarrow \Phi_o - \alpha\nabla_{\Phi_o}\mathcal{L}(\tau_k) \tag{28}$$

Unlike the existing gradient-based meta-learning methods, such as model-agnostic meta-learning (MAML) [10], first-order MAML (FOMAML) [10] and Reptile [22], which treat all parameters and tasks equally, DCMR groups the parameters and updates only the prediction module to follow the optimal solution manifolds of new tasks. The purpose of this is to reproduce only the feature modeling of replay tasks and weaken the disturbance to the prediction module

to achieve more general representation learning. Therefore, DCMR can avoid overfitting the remaining tasks and enhance the change of new tasks on the meta model.

DCMR is deployed in two parts, online and offline (Figure 3). Except for online serving, the initial training, user-incremental updates, and memory reconstruction are all performed offline, which will not increase the service delay of the system but only affect its real-time performance (rapid response to incremental information). Next, we analyze the time complexity of this stage based on three meta-optimization algorithms: second-order meta-learning, first-order meta-learning and DCMR. For simplicity, we assume that the representation module contains k feature matrices with a size of $a \times b$, the prediction module is an l-layer MLP, and each layer contains n neurons. Normally, the reconstruction of the replay task memory requires only sampling and distance calculation, which has a much lower time consumption than backpropagation. Therefore, it is performed in parallel with user-incremental update and is completed first, and it will not affect the total time consumption. For second-order meta-learning (such as MAML), due to the calculation of the Hessian matrix, its time complexity is expressed as $O(k \cdot a^2 \cdot b^2 + l \cdot n^3)$. For the previous first-order meta-learning (such as Reptile), since the global update does not need to calculate the second step, its time complexity is consistent with the traditional training method, denoted as $O(k \cdot a \cdot b + l \cdot n^2)$. For DCMR, since the prediction result y° of the meta model is calculated at the beginning of local update, the distillation loss does not bring additional computational burden to the model, so we ignore it. Consider the time complexity of the new task as $O(k \cdot a \cdot b)$ and the time complexity of the replay task as $O(k \cdot a \cdot b + l \cdot n^2)$. Then, the total time complexity of DCMR is $O(k \cdot a \cdot b + h \cdot l \cdot n^2)$, where h is the proportion of new tasks and $h < 1$. In summary, DCMR has a lower time complexity than the previous meta-update methods. Furthermore, DCMR can speed up calculations through task parallelism during local updates.

4 EXPERIMENTS

In this section, we seek to address the following key research questions (RQs):

- **RQ1** : How good is the performance of DCMR in comparison to that of the SOTA methods? (Section 4.3)
- **RQ2** : Compared with other incremental update strategies applicable to MLRSs, does DCMR achieve higher computational efficiency? (Section 4.4)
- **RQ3** : Are the components designed for incremental updates in DCMR necessary? (Section 4.5)
- **RQ4** : How well does DCMR generalize when we tune its hyperparameters? (Section 4.6)

4.1 Datasets

We conduct experiments on three widely used benchmark datasets: **MovieLens-1M**[4], **Bookcrossing**[5] and **MovieLens-20M**[6], all of which contain a large number of user ratings of movies or books. To evaluate DCMR in different scenarios, we treat MovieLens-1M and Bookcrossing as explicit feedback datasets with contents of users and items and transform MovieLens-20M into an implicit feedback dataset without any contents.

For MovieLens-1M, we process it following [16] and sort users according to their first rating time to build the user stream. For Bookcrossing, following the data processing method in [7], we clean it up and add sequence information to it. For MovieLens-20M, following [8], we regard all movies rated by each user as positive items for this user and treat the ID of each user or item as its unique feature. As suggested by previous works [16], we select only the users with rating counts between 20 and 100 as the cold-start users in these datasets. In particular, we adopt negative sampling in MovieLens-20M to supplement all users with the same number of negative items as positive ones.

During the test, for the first two datasets, we use the last 10 rating records of each user as the query set and evaluate the performance of the model on them, and the remaining records are treated as the support set. For MovieLens-20M, we regard the last 10 positive and 10 negative samples of each user as the query set.

4.2 Setup

4.2.1 Evaluation Metrics. Following the settings in [25], all datasets are split into 30 periods, and each period contains the same number of users (the last period is slightly more or less). The first 10 periods are used to warm up the model (initial training), and the incremental update starts from the 11th period. We use the 11th to 15th periods as the validation set to select appropriate hyperparameters, and we report the average performance on the query sets in the last 15 periods.

In the explicit feedback scenarios, we adopt the mean average error (MAE) and normalized discounted cumulative gain (NDCG)@5 as the evaluation metrics. In the implicit feedback scenario, we adopt recall@10 and NDCG@10.

4.2.2 Compared Models. We compare DCMR with three groups of models: **Group 1**: Traditional incremental strategies, including **FR**, **SR** and **FT**; **Group 2**: MLRSs, including **Reptile** [22], **MeLU** [16] and **MAMO** [7]; and **Group 3**: Incremental RSs, including **SPMF** [38], **SML** [44], **IncCTR** [40] and **ASMG** [25].

Group 1: As described earlier, this group of methods are all based on a basic first-order meta-optimization method (Reptile). The purpose of this group is to evaluate the difference between the update strategies of DCMR and traditional methods. **FR** saves all data and retrains the model at each update stage. **SR** maintains a replay task memory similar to that of DCMR and merges it with new data as training data in each update stage. **FT** only keeps the meta model after each update and fine-tunes it with the new data in the next stage.

Group 2: The methods in this group are MLRSs without user-incremental updates, which are only trained in the warm-up stage. **Reptile** is a classic first-order meta-learning method, and we transfer it to RSs. **MeLU** is a recommendation algorithm based on a second-order meta-optimization method (MAML). We modify its input and prediction parts to fit MovieLens-20M. **MAMO** utilizes two additional memory matrices to improve the adaptation of MAML for more personalized recommendations. Since the attention values in MAMO are calculated via the content information, we do not report its performance on MovieLens-20M.

[4]https://grouplens.org/datasets/movielens/1m/
[5]http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[6]https://grouplens.org/datasets/movielens/20m/

Group 3: This group contains four incremental RSs, which can be continuously updated through data streams. Since these methods are not MLRSs, we use all the support sets in the test stage as the training data in the previous stage. **SPMF** is a sample-based approach for incremental recommendation, which mixes the selected samples with new data to update the model. **SML-MLP** is a model-based framework used to obtain the incremental RS. We adopt MLP as the base model, which performs better according to the reported results in [25]. **IncCtr** is an incremental click-through rate (CTR) prediction model based on knowledge distillation. **ASMG** is another model-based framework for the incremental updating of RSs. As suggested in the original paper, it should be trained in two stages: basic model warm-up and metagenerator training. For a fair comparison, we warm up the basic model on the first 5 periods, train the metagenerator on the 6th to 10th periods, and use the 11th to 15th periods as the validation set. SPMF and IncCTR utilize specially designed loss functions that are not suitable for explicit feedback scenarios (MovieLens-1M and Bookcrossing), so we only report their performance on MovieLens-20M.

4.2.3 Hyperparameters. For fairness, we set the same network architecture for all models. In the representation module, the embedding size of each feature is fixed to 32. The prediction module is set to a $[\rightarrow 64 \rightarrow 64 \rightarrow 1]$ MLP with a rectified linear unit (ReLU). For all MLRSs, the local and global learning rates are determined by grid searches in the range $[0.1, 0.001, 0.0001, 0.00001]$, each batch contains 32 tasks, and the meta-adaptation steps are searched in the range $[1, 2, 3, 4, 5]$. The other model-specific hyperparameters are set according to the performance on validation sets or authors' suggestions. For DCMR, α, β, λ, η (for DCMR+), M, the batch size and the update epochs are set to 0.01/0.01/0.01, 0.1/0.1/0.05, 0.001/0.001/0.001, 0.2/0.2/0.5, 400/800/8000, 32/32/256 and 5/5/10 on the three datasets, respectively. We do not limit the value of L but only discuss this point in the ablation experiments.

4.3 Performance Comparison (RQ1)

The performance of the compared models is reported in Table 1. **S-DCMR** denotes the DCMR based on the **s**econd-order meta-optimization framework (MAML). From these results, we have the following observations:

- The MLRSs outperform the traditional incremental RSs in general, even if some of them are no longer updated during the testing phase. This proves the excellent effect of meta-learning in personalized recommendation (especially in cold-start scenarios). On MovieLens-1M and Bookcrossing, the MLRSs are more stable than the methods in Group 3, but there is no significant improvement on MovieLens-20M. One possible reason is that MLRSs can better utilize rich contextual information to adapt to new users for more stable performance.
- In **Group 1**, the performance of FT on the first two datasets is significantly behind that of FR and SR, but on MovieLens-20M, the opposite is true. This may be because compared to MovieLens-1M and Bookcrossing, MovieLens-20M has a longer time span, that is, it reflects a longer-term fashion trend. Therefore, in a more stable environment, FR and SR can effectively avoid catastrophic forgetting through task replay, thereby making full use of the preference information of old users. However, in a nonstationary environment, FR and SR cannot quickly adapt to the latest trends because they cannot distinguish between new and old users. In contrast, FT can overcome this issue. In addition, the performance of SR is always between that of FR and FT, which proves that it can weigh the advantages of these two strategies, but its performance is not good due to overfitting the replay tasks. This also confirms our conjecture: during the incremental update, the model should be able to balance the trade-offs between old and new users and avoid overfitting.
- In **Group 2**, the second-order meta-learning methods (MeLU and MAMO) are generally better than the first-order one (Reptile). An important reason is that the first-order method is more sensitive to parameters and data, and it cannot maintain stable generalization in this drastically changing data environment. However, there is no significant difference between S-DCMR and DCMR, especially DCMR, which even surpasses S-DCMR on MovieLens-20M. This may be because the special design for incremental updates offsets the disadvantage of the first-order method in generalization.
- In **Group 3**, the model-based methods (ASMG and SML-MLP) consistently outperform SPMF and IncCTR, which shows that it is feasible to utilize additional memory components to extract information from the previous models. In other words, the previous models already contain enough knowledge to indicate the preference of the old users. This also proves the feasibility of activating the knowledge about old users in the initial meta model through a small number of replay tasks in DCMR. Moreover, because ASMG considers multistage models, its performance is significantly better than that of SML-MLP, which only utilizes the model from the previous stage. In DCMR, we utilize multistage information by continuously reconstructing the replay task memory.
- DCMR significantly outperforms all the baselines, which demonstrates its significant advantages in user stream scenarios. DCMR+ achieves stronger results on MovieLens-1M, which shows the potential advantages of the momentum-based dynamic fusion parameter in maintaining model stability in long-term data streams.

Figure 4 shows the performance of the models on a data stream split by a certain length of time. We regard three months as one period and report the results of the last 15 periods. DCMR always performs better than the other models. When the amount of data in different periods changes drastically, the DCMR-based methods can be almost unaffected and remain stable.

4.4 Computational Efficiency (RQ2)

In addition to the improvement of performance, another goal of DCMR is to speed up the user-incremental updates of MLRSs. We compared the time consumption of FR, SR, DCMR and S-DCMR in several update periods on MovieLens-1M. The experiment was conducted on a Linux server with a GPU (NVIDIA TITAN X) and CPU (Intel Core i7-5930K), and the results are reported in Table 2.

Table 1: Experimental results of different models on MovieLens-1M, Bookcrossing and MovieLens-20M. To study the stability of the models, the results are reported as ± std, and these values are computed from the last 15 periods of model updating. For Recall@k and NDCG@k, larger values are better, and the opposite is true for MAE. The best results are marked in bold, and the second-best results are underlined. "I/U" indicates the RS with incremental updates, and "R/" indicates a replay-based method.

Model	I/U	MLRS	R/	MovieLens-1M		Bookcrossing		MovieLens-20M	
				MAE↓	NDCG@5↑	MAE↓	NDCG@5↑	Recall@10↑	NDCG@10↑
FR	✓	✓	✓	0.7523±.0240	0.8117±.0154	1.6939±.0283	0.8191±.0115	0.6402±.0187	0.6937±.0175
SR	✓	✓	✓	0.7688±.0179	0.7955±.0099	1.8508±.0324	0.8078±.0205	0.6876±.0286	0.6772±.0150
FT	✓	✓		0.7984±.0190	0.7864±.0180	2.1103±.0358	0.7423±.0196	0.7294±.0318	0.7004±.0119
Reptile		✓		0.7665±.0116	0.8023±.0074	1.7296±.0255	0.8010±.0072	0.7059±.0194	0.6678±.0127
MeLU		✓		0.7622±.0079	0.8051±.0068	1.6953±.0230	0.8075±.0078	0.7311±.0225	0.6880±.0143
MAMO		✓		0.7583±.0142	0.8144±.0070	1.6825±.0208	0.8277±.0088	-	-
SPMF	✓		✓	-	-	-	-	0.6145±.0117	0.6125±.0178
SML-MLP	✓	○		0.8191±.0236	0.7656±.0137	1.9110±.0518	0.7868±.0135	0.6881±.0163	0.6348±.0089
IncCTR	✓			-	-	-	-	0.6289±.0138	0.6096±.0175
ASMG	✓	○		0.8026±.0212	0.7939±.0126	1.8465±.0466	0.7861±.0142	0.6973±.0158	0.6533±.0103
S-DCMR	✓	✓	✓	**0.7298**±.0086	**0.8368**±.0059	**1.6415**±.0215	0.8258±.0077	0.7160±.0157	0.6845±.0099
DCMR	✓	✓	✓	0.7371±.0132	0.8281±.0070	1.6663±.0203	**0.8344**±.0083	**0.7603**±.0163	**0.7226**±.0074
DCMR+	✓	✓	✓	0.7358±.0114	0.8355±.0074	1.6624±.0243	0.8332±.0077	0.7394±.0174	0.7042±.0082

○ This is a generalized meta-learning-based method, but not the narrow MLRS defined in this paper.

The following conclusions can be drawn: 1) The time consumption of FR increases with the number of update periods, and it is always higher than that of other methods. This is mainly due to the accumulation of previous data during the update periods. 2) The update of S-DCMR is slower than that of SR and DCMR, which shows that the first-order method has higher computational efficiency than the second-order method. 3) Compared with that of SR, the update time of DCMR is slightly shorter. One possible reason is that the limitation imposed by DCMR on the parameters of the replay tasks speeds up the computation.

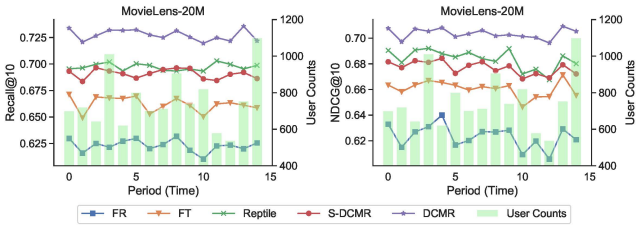


Figure 4: The performance of the model that is updated with fixed time periods (3 months).

4.5 Ablation Study (RQ3)

In this section, we compare the performance of DCMR and its three ablation models. **DCMR-D** removes the **d**istillation losses of replay tasks, **DCMR-P** updates all **p**arameters for replay tasks in the local updates, and **DCMR-B** adopts distillation losses in both replay and new tasks. Figure 5 reports the results of DCMR and its variants on the three datasets.

We obtain the following observations from this experiment: 1) DCMR performs best on all datasets, which proves the effectiveness of distillation loss and the special parameter update strategy. 2) Among the ablation models, DCMR-D is worse than DCMR-P and DCMR-B and even worse than SR in some cases. This shows that the distillation loss has a higher priority in all components, and the parameter update strategy does not work well without distillation loss.

Table 2: Updating time (seconds) in the test periods on MovieLens-1M. For a more intuitive presentation, FR is only trained for 5 epochs, as is DCMR, at each period, and FR has worse performance than that reported in Table 1.

Periods	16	19	22	25	28	30
FR	698.5	768.7	833.0	871.7	903.2	1014.7
SR	81.4	82.3	79.8	84.9	82.3	85.8
DCMR	80.0	79.2	81.3	81.1	80.6	81.8
S-DCMR	99.7	99.8	101.5	104.0	103.4	97.2

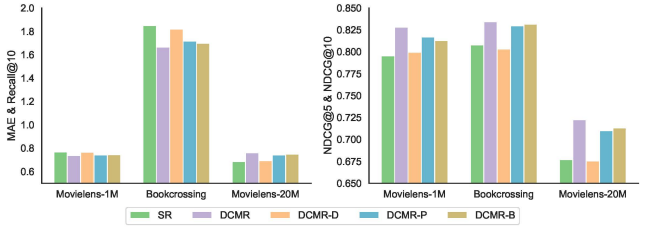


Figure 5: The performance of DCMR and its three ablation models.

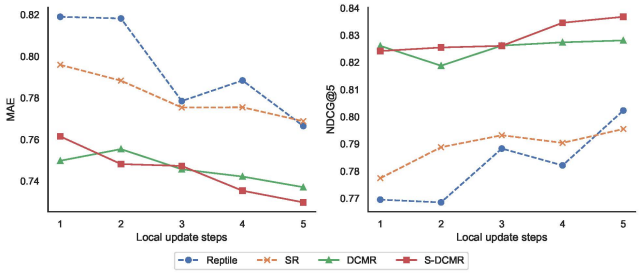


Figure 6: Sensitivity to the local update steps.

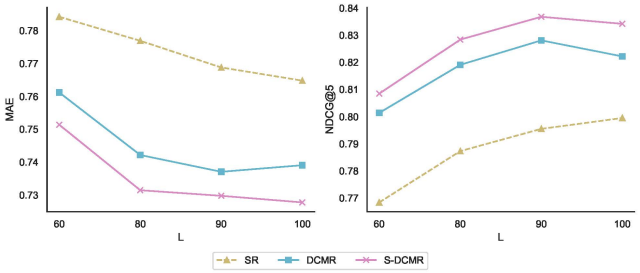


Figure 7: Sensitivity to the memory size.

4.6 Hyperparameter Analysis (RQ4)

Here, we investigate the sensitivity of DCMR with respect to two main hyperparameters: the local update steps and L. Since similar trends are observed on the three datasets, we only report the results on MovieLens-1M.

As shown in Figures 6 and 7, DCMR and S-DCMR always perform better than Reptile and SR. For the update steps, our models are not very sensitive, so we can appropriately reduce them to improve the efficiency of the models. For the maximum number of items selected for each user, a larger size does not bring a significant improvement, so we can choose an appropriate size to reduce the memory consumption.

5 RELATED WORK

5.1 MLRSs

MLRSs are a mainstream RS paradigm, and they achieve excellent results in various scenarios. In a broad sense, MLRSs cover many categories, such as metric-based methods [46], model-based methods [9, 18], and gradient-based methods [16, 20, 34]. However, here, we specifically refer to the gradient-based method that has recently become popular.

Unlike traditional RSs, MLRSs regard making recommendations for one user as a separate task, and this can provide more personalized services. In general, MLRSs first learn a set of initial metaparameters for all users and then obtain user-specific models through a few steps of task-aware adaptation for cold-start users.

Among the existing MLRSs, MAML [10] is a popular general second-order meta-learning framework, and most MLRSs are developed based on it. MeLU [16] is a classic MAML-based second-order MLRS that regards cold start as a few-shot problem and proposes dividing data into different tasks according to different users to adapt to the setting of meta-learning. MetaHIN [20] introduces heterogeneous information networks into the MLRS to strengthen the representation of users and items and achieves better performance. MAMO [7] adopts two memory matrices in the MLRS: a feature-specific matrix that guides the model with personalized parameter initialization and a task-specific matrix for fast prediction of user preferences. PAML [43] utilizes a task-adaptive learning rate to improve the user-specific models' performance for major and minor users simultaneously.

The user-incremental update framework we proposed in this paper has excellent applicability and flexibility, which can be applied to most MLRSs, including the above.

5.2 RSs with Incremental Updates

Unlike the training-serving framework followed by most RSs [14], real-world data always enter the system in the form of streams, which requires that RSs can be updated at any time, that is, can perform incremental updates.

Inspired by experience replay in continuous learning [28], early research adopted random sampling to construct a sample memory for the subsequent incremental updates [6]. After that, various studies focused on designing a better heuristic strategy to maintain this memory, such as order-based methods [4, 45] and forgetting-based methods [13, 26]. However, one limitation of these methods is that the small number of samples in memory is insufficient to reflect the distribution of all historical samples. To address this problem, recent model-based approaches tend to extract knowledge from historical models [21, 42]. IncCTR [40] regards the previous models as teacher models and trains the student models for the subsequent periods based on knowledge distillation. SML [44] employs a CNN-based transfer module to leverage the previous model in the next periods. ASMG [25] considers long-term sequential patterns to extend SML to generate a better model for future serving.

The existing methods are highly versatile, but they are not compatible with MLRSs. Our DCMR is a user-incremental update framework specially customized for MLRSs. Simultaneously, unlike SML and ASMG, DCMR does not need to warm up additional parameter generation components, so it is more practical.

6 CONCLUSION

In this work, we propose a deployable and continuable MLRS framework (DCMR) with fast user-incremental updates. DCMR maintains a fixed-size replay task memory and utilizes it to activate the old user preference distribution contained in previous meta-models. To achieve this goal, we specifically design two distillation loss functions for two common feedback modes in RSs and balance the trade-off of new and old users through task-specific parameter update strategies. Additionally, DCMR can use a momentum-based dynamic fusion parameter to adjust each update period for a more stable procedure. Finally, DCMR is developed via a first-order meta-optimization strategy, which has higher computational efficiency than traditional second-order strategies. It is also compatible with most existing MLRSs. We conduct sufficient experiments on three benchmark datasets to evaluate and analyze its rationality, effectiveness, versatility and efficiency.

ACKNOWLEDGMENTS

The authors are grateful for the support of the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, National Key R&D Program of China [No.2021ZD0112501 and No.2021ZD0112502], National Natural Science Foundation of China [No. 61972174 and No.62172187], the Science and Technology Planning Project of Guangdong Province [No. 2020A0505100018], Guangdong Universities' Innovation Team Project [No.2021KCXTD015], Guangdong Key Disciplines Project [No.2021ZDJS138] and the Tencent Rhino-Bird Research Program. Fausto Giunchiglia's work is funded by National Project PRIN 2017 "Delphi".

REFERENCES

[1] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. 2011. Analyzing user modeling on twitter for personalized news recommendations. In *international conference on user modeling, adaptation, and personalization*. Springer, 1–12.

[2] Homanga Bharadhwaj. 2019. Meta-learning for user cold-start recommendation. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[3] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420* (2018).

[4] Chen Chen, Hongzhi Yin, Junjie Yao, and Bin Cui. 2013. Terec: A temporal recommender system over tweet stream. *Proceedings of the VLDB Endowment* 6, 12 (2013), 1254–1257.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[6] Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. 2012. Real-time top-n recommendation in social streams. In *Proceedings of the sixth ACM conference on Recommender systems*. 59–66.

[7] 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*. 688–697.

[8] 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*. 2895–2904.

[9] Xidong Feng, Chen Chen, Dong Li, Mengchen Zhao, Jianye Hao, and Jun Wang. 2021. CMML: Contextual Modulation Meta Learning for Cold-Start Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 484–493.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. PMLR, 1126–1135.

[11] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. 2018. Conditional neural processes. In *International Conference on Machine Learning*. PMLR, 1704–1713.

[12] Asnat Greenstein-Messica and Lior Rokach. 2018. Personal price aware multi-seller recommender system: Evidence from eBay. *Knowledge-Based Systems* 150 (2018), 14–26.

[13] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1569–1577.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[15] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 831–839.

[16] 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*. 1073–1082.

[17] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[18] Xixun Lin, Jia Wu, Chuan Zhou, Shirui Pan, Yanan Cao, and Bin Wang. 2021. Task-adaptive Neural Process for User Cold-Start Recommendation. In *Proceedings of the Web Conference 2021*. 1306–1316.

[19] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017), 6467–6476.

[20] 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*. 1563–1573.

[21] Fei Mi, Xiaoyu Lin, and Boi Faltings. 2020. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 408–413.

[22] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[23] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. PinnerSage: multi-modal user embedding framework for recommendations at pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2311–2320.

[24] Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. 2019. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 695–704.

[25] Danni Peng, Sinno Jialin Pan, Jie Zhang, and Anxiang Zeng. 2021. Learning an Adaptive Meta Model-Generator for Incrementally Updating Recommender Systems. In *Fifteenth ACM Conference on Recommender Systems*. 411–421.

[26] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. Gag: Global attributed graph neural network for streaming session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 669–678.

[27] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. 2020. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13588–13597.

[28] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[29] Steffen Rendle and Lars Schmidt-Thieme. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*. 251–258.

[30] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. 2018. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682* (2018).

[31] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*. PMLR, 1842–1850.

[32] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 253–260.

[33] Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175* (2017).

[34] 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*. 1177–1186.

[35] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems.. In *NIPS*. 4957–4966.

[36] Jianling Wang, Kaize Ding, and James Caverlee. 2021. Sequential Recommendation for Cold-start Users with Meta Transitional Learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1783–1787.

[37] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2467–2475.

[38] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming ranking based recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 525–534.

[39] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[40] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2020. A Practical Incremental Method to Train Deep CTR Models. *arXiv preprint arXiv:2009.02147* (2020).

[41] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 374–382.

[42] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. 2020. Graphsail: Graph structure aware incremental learning for recommender systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2861–2868.

[43] Runsheng Yu, Yu Gong, Xu He, Bo An, Yu Zhu, Qingwen Liu, and Wenwu Ou. 2020. Personalized Adaptive Meta Learning for Cold-start User Preference Prediction. *arXiv preprint arXiv:2012.11842* (2020).

[44] 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*. 1479–1488.

[45] Yan Zhao, Shoujin Wang, Yan Wang, and Hongwei Liu. 2021. Stratified and time-aware sampling based adaptive ensemble learning for streaming recommendations. *Applied Intelligence* 51, 6 (2021), 3121–3141.

[46] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. 2020. Cold-start Sequential Recommendation via Meta Learner. *arXiv preprint arXiv:2012.05462* (2020).

[47] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[48] Yu Zhu, Jinghao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2019. Addressing the item cold-start problem by attribute-driven active learning. *IEEE Transactions on Knowledge and Data Engineering* 32, 4 (2019), 631–644.