A Dynamic Meta-Learning Model for Time-Sensitive Cold-Start Recommendations

Krishna Prasad Neupane, Ervine Zheng, Yu Kong, Qi Yu

Rochester Institute of Technology
{kpn3569, mxz5733, yu.kong, qi.yu}@rit.edu

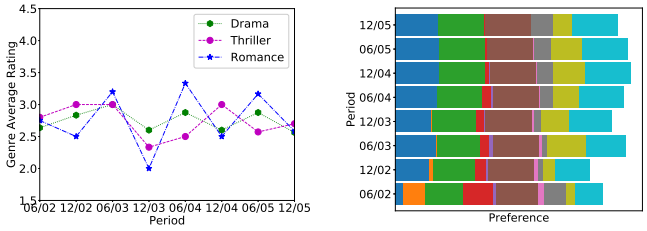
Abstract

We present a novel dynamic recommendation model that focuses on users who have interactions in the past but turn relatively inactive recently. Making effective recommendations to these *time-sensitive cold-start users* is critical to maintain the user base of a recommender system. Due to the sparse recent interactions, it is challenging to capture these users' current preferences precisely. Solely relying on their historical interactions may also lead to outdated recommendations misaligned with their recent interests. The proposed model leverages historical and current user-item interactions and dynamically factorizes a user's (latent) preference into time-specific and time-evolving representations that jointly affect user behaviors. These latent factors further interact with an optimized item embedding to achieve accurate and timely recommendations. Experiments over real-world data help demonstrate the effectiveness of the proposed time-sensitive cold-start recommendation model.

Introduction

Recommender system has long been used as an effective means to improve user experience and to provide personalized recommendations in diverse fields such as media, entertainment, and e-commerce. [25, 33]. One effective way of recommendation is via Collaborative Filtering (CF) [9, 24], which recommends items based on similar users' preferences. CF assumes that users who had similar interactions with some items in the past are likely to have the same preference on other items, and leverages the observed user-item interactions to make predictions for the missing parts, which indicate the potential items of interest to users. Matrix Factorization (MF) is one commonly used CF technique that exploits user and item latent factors to capture their inherent attributes. Most existing MF methods model user preferences and item attributes as static factors [14, 16]. Some recent efforts consider the dynamic changes in the user-item interactions by modeling user preferences as shifting latent factors over time, allowing them to provide more timely recommendations [3, 10, 25].

When user-item interactions are very sparse, making accurate recommendations based on limited information is highly challenging, which is usually referred to as the cold-start problem. A viable solution to handle the cold-start problem is to utilize extra side information such as item descriptions, user profiles [27], and social relationships [35]. While various side information is widely available for items (i.e., movies, songs, and books), the user-related side information is typically very scarce as acquiring users' personalized information may be practically difficult due to privacy issues. Several recent works adopted meta-learning approaches for few-shot learning in the recommender systems to alleviate the cold-start problem [28, 17, 5]. Meta-learning aims to learn global knowledge from the historical information of many similar tasks (users) and then provide quick adaptation for a new task (user) with limited interaction information. Although the meta-learning approaches show promising results [28, 17], they are primarily designed for static settings and hence not effective in providing *timely recommendations that best reflect users' current interests*.



(a) (b)

Figure 1: For User (ID:2181970): (a) shows dynamic changes of ratings for three genre types over time (b) demonstrates how user's latent preferences evolve over time.

Figure 1a shows that the average movie ratings for three genre types of an example user from the Netflix dataset, which oscillate significantly from 2002 to 2005. The average rating indicates an overall satisfaction on each type of genre items that the user interacts with for each period, and serves as an indicator for the change of users' preference. Those changes are usually caused by the interplay of two contributing factors. First, a user's preference over different types of items (e.g., movie or music) may change over time, which we refer to as the *time-evolving factor*. Second, a user's behavior in a specific period may vary significantly from other periods due to the impact of money/time bud-

get or other external causes, which we refer to as the *time-specific factor*. Given sufficient user interactions over time, both factors could be effectively learned to provide accurate and timely recommendations. However, in practice, many users may have interactions in prior periods but become largely inactive with few interactions in recent periods for various reasons. We define these users as the *time-sensitive cold-start users* due to scarce recent interactions. Solely relying on the historical interactions of these users may lead to outdated recommendations that do not match their recent interests. Furthermore, the limited recent interactions pose a user cold-start problem for the current period that makes existing CF-based techniques less effective.

The main challenge with these time-sensitive cold-start users is to simultaneously capture their most recent interests and evolving preferences, which are keys to achieve accurate and timely recommendations. In this paper, we focus on this special *time-sensitive cold-start problem*, which is critical for a recommender system to maintain its user base. We propose to dynamically factorize a user's (latent) preference into time-specific and time-evolving representations in order to capture the time-specific and time-evolving factors from both the historical and current user-item interactions. For example, the Netflix dataset consists of a user's interactions with a large movie set in the form of user ratings. The variation of movie ratings from the same user may be affected by the change of user preference, rating criteria, or other (unknown) factors. In addition, a user's preference for different genres may evolve over multiple periods, as shown in Figure 1b, which corresponds to the proportion of different factors in the time-evolving representation discovered by our proposed model.

The proposed model consists of two distinct modules: a *meta-learning module* and a *recurrent module*. The meta-learning module aims to capture time-specific latent factors through limited interaction data by leveraging the shared knowledge learned from other users. The recurrent module aims to capture time-evolving latent factors by nesting a recurrent neural network, and it can be jointly optimized with the meta-learning module through the model-agnostic meta-learning approach [7]. Finally, we seamlessly integrate the two modules by merging the time-specific and time-evolving factors to form the user representation. This user representation further interacts with an item embedding (which is also optimized during model training) to provide the final recommendations. Our experimental results clearly show that the proposed model makes timely recommendations that closely resemble the dynamically changed user ratings as a result of effectively integrating the complementary factors capturing the user preferences.

The **main contributions** of this paper are five-fold: (i) the **first work** to formulate the time-sensitive cold-start problem that is critical to maintain the user base of a recommender system; (ii) a **novel integrated recommendation framework** to model sparse dynamic user-item interactions and extract time-evolving and time-specific factors of user preferences simultaneously; (iii) a **time-sensitive meta-learning module** to effectively handle user cold-start problems by leveraging knowledge shared across multiple users from the current recommendation period to adapt to any specific user's case using limited interaction information, (iv) a **time-evolving recurrent module** to effectively capture the gradual shift of users' preferences over time, and (v) an **integrated training process** that combines these two models to simultaneously learn time-specific and time-evolving factors and optimizes the item embedding.

We conduct extensive experiments over multiple real-world datasets and compare with representative state-of-art dynamic and meta-learning-based recommender systems to demonstrate the effectiveness of the proposed model.

Related Work

Matrix Factorization Models. Matrix factorization is a commonly used collaborative filtering approach that characterizes users and items through the latent factors inferred from the item rating patterns [16]. Using singular value decomposition (SVD) for recommendation is popularized by the *Simon Funk* [8] in the Netflix prize competition and a probabilistic version is introduced by [19]. SVD is extended to SVD++ [14] for processing the implicit feedback.

The above static models do not include the important temporal information, which should be considered for analyzing user-item interactions in a time-sensitive setting. Dynamic matrix factorization has been developed to address the issue, which allows latent features to change with time. Some works introduce time-specific factors, such as timeSVD++, which uses additive bias to model user-related temporal changes [15]. There have been works that employ Gaussian state-space models to introduce time-evolving factors with a one-way Kalman filter [25, 10]. To handle implicit data, Sahoo et al. propose an extension to the hidden Markov model [22], where clicks are drawn from a negative binomial distribution and Charlin et al. [3] introduces a Gaussian state-space model with the Poisson emission. However, matrix-factorization-based algorithms may suffer from limited expressive power and may not capture the complex nature of user-item interactions. Instead, our model captures both dynamic and static user preferences.

Deep Learning Models. Recent works in recommender systems [4, 11, 37] utilize deep learning to provide better recommendations. Cheng et al. [4] propose to jointly train wide linear models and deep neural networks to combine the benefits of memorization and generalization. Similarly, DeepFM [11] integrates the power of deep learning and factorization machines models to learn low- and high-order feature interactions simultaneously from the input. DIEN [37] formulates interest evolution network as a deep learning model to capture latent temporal interests and evolving interests for better recommendations. These models are very sensitive to the features and might need important features information and large datasets while training, which is not necessary for our model.

Graph-Based Models. Another popular line of recommendation systems is graph-based models. A graph captures high-order user-item interactions through an iterative process to provide effective recommendations [12]. Users and

items are represented as a bipartite graph in [2] and links are predicted to make recommendations. Similarly, a graph-based framework called Neural Graph Collaborative Filtering (NGCF) [31] explicitly encodes the collaborative signal in the form of high-order connectivities in a user-item bipartite graph via embedding propagation. However, these methods are unable to capture long-term user preferences or deal with cold-start problems.

Sequential Models. Sequential models understand the sequential user behaviors via user-item interactions, and model the evolution of users' preferences and item popularity over time [30, 6]. Tang et al. [26] utilizes convolutional sequence embedding to capture union level and point level contributions of historical items via horizontal and vertical filters and provides top-N sequential recommendations. Similarly, Kang et al. [13] introduce a self-attentive mechanism to handle both long and short-term user preferences in a sequential setting. Sequential models focus on users' evolving preferences (i.e., recent interactions) but largely neglect long-term users' preferences. In contrast, our model dynamically captures long-term time-evolving preference via RNN and time-specific users preferences through meta-learning to alleviate cold-start problems.

Meta-learning Models. The user-item interaction data is usually sparse because a user may only interact with a few items within the large item pool. In such cases, making recommendations can be viewed as a few-shot learning problem. Meta-learning [23, 1] is recently becoming a popular few-shot learning approach that learns from similar tasks and can generalize quickly and efficiently for the few-shot unseen new tasks. Finn et al. [7] propose a model-agnostic meta-learning model that learns global parameters from a large number of tasks and performs as a good generalization on a new task that has few samples utilizing the few steps of gradients. To address the cold-start problem in item recommendation, Vartak et al. [28] introduce a meta-learning strategy that focuses on the items-cold-start problem to recommend cold-start items considering that items arrive continuously in the Twitter Timeline. On the contrary, our model focuses on recommendations for the time-sensitive, cold-start users by capturing users' preferences over time. Similarly, recent work based on meta-learning is done to estimate user preferences in [17] and scenario-specific recommendation in [5]. Both works use the information of users and items to generate user and item embeddings. Meta-learning is specifically utilized to learn heterogeneous information networks to alleviate cold-start problems [18]. Similarly, meta-learning approachs are utilized in graphs [32, 34] to quickly adapt to new sub-graph that alleviate the cold-start problem. Also, meta-learning is applied in click-through rate (CTR) prediction [21] where desirable embeddings for the new ads are generated via meta-learner. These embedding methods are designed for a static setting, making them not applicable to learn user preferences evolving over time. In our work, we simultaneously learn time-evolving and user-specific preferences to provide accurate and timely recommendations for time-specific cold-start users with very limited recent interaction data.

Table 1: Summary of Notations

Notation	Description
u, i	user and item
z_i, e_i	item i's original encoding and embedding
$\hat{r}^t_{(u,i)}$, $r^t_{(u,i)}$	predicted and ground truth scores for user u on item i at period t
u^t_{ts}, u^t_{te}	time-specific and time-evolving factors of user u at period t
θ^t, θ^t_u	meta and user-specific parameters of time-specific module at period t
ω	parameter of time-evolving module
$\mathcal{S}^t_u$, $\mathcal{Q}^t_u$	support and query sets in task corresponding to user u at period t
D^t_u	items interacted with user u at period t

The Dynamic Meta-Learning Recommendation Model

Problem Settings. We propose a dynamic recommendation model, where the input data is represented as $\{\mathcal{U}, \mathcal{I}, \mathcal{H}\}$, $\mathcal{U}$ is the user set, $\mathcal{I}$ is the item set, and $\mathcal{H}$ is the set of time periods. A time period $t \in \mathcal{H}$ defines a particular time interval where the interactions for user u with the item i are aggregated based on timestamps. All major symbols are summarized in Table 1. We perform recommendation in each time period with a recommendation function as

$$f^t_{\theta^t_u,\omega}(i) = \hat{r}^t_{(u,i)} \qquad \forall u \in \mathcal{U}, i \in \mathcal{I}, t \in \mathcal{H} \tag{1}$$

where $\hat{r}^t_{(u,i)}$ is the recommended score for item i assigned by user u at period t, θ^t_u is user latent factor at time t, and ω is the parameter of the recurrent neural network module. The goal of a recommender system is to predict the recommendation scores that can accurately capture a user's true preference on items over time so that the recommended items are likely to be adopted by the users.

We formulate dynamic recommendations as a few-shot regression problem in the meta-learning setting. Users are *dynamically* partitioned into *meta-train* and *meta-test* sets based on their interactions in the current recommendation period t. In particular, the meta-train user set includes users with sufficient interactions, while the meta-test user set includes time-sensitive cold-start users who have only a few interactions in the current time period. Details for the train-test user splits are discussed in the experiment section. We consider a distribution over tasks $P(\mathcal{T})$, and each user is represented as a few-shot regression task $\mathcal{T}^t_u$ sampled from the given task distribution. In general, a task includes a *support set* $\mathcal{S}^t_u$ and a *query set* $\mathcal{Q}^t_u$. The support set includes a user's interactions at period t where k is interpreted as the number of shots (i.e., interactions). The query set includes the rest interactions of this user at period t.

$$\begin{aligned} \mathcal{T}^t_u \sim P(\mathcal{T}): \quad & \mathcal{S}^t_u = \{(u, i_j), r^t_{(u,i_j)}\}_{j=1:k}, \\ & \mathcal{Q}^t_u = \{(u, i_j), r^t_{(u,i_j)}\}_{j=k+1:N_t} \end{aligned} \tag{2}$$

where N_t is the number of items a user interacted with at period t and $r^t_{(u,i_j)}$ represents label (i.e. rating or count) from user u to item i_j. We adopt episodic training [29], where the

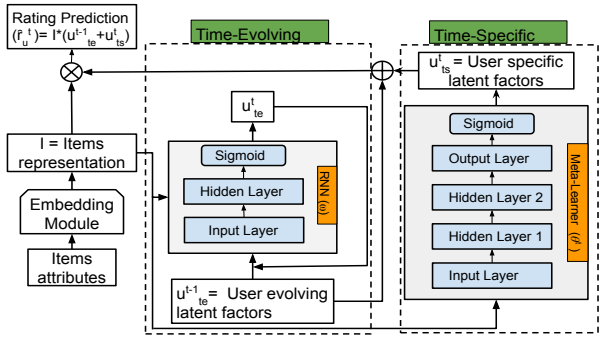


Figure 2: The proposed model captures time-evolving factors via a recurrent network module and time-specific factors through a meta-learning module.

training task mimics the test task for efficient meta-learning. The support set $\mathcal{S}$ in each episode works as the labeled training set on which the model is trained to minimize the loss over the query set $\mathcal{Q}$. The training process iterates episode by episode until convergence.

Model Overview

To leverage item information such as text descriptions, our model generates an initial item representation (I) using an embedding matrix $E \in \mathbb{R}^{d \times m}$, where m is the dimension of input item attributes, and d is the dimension of embedding. The embedding is generated from item attributes following [4, 17]. An item i is first encoded as a binary vector $z_i \in \mathcal{R}^m$, where the corresponding index of item attributes is set to 1 and 0 otherwise. The binary vector is then transformed using the embedding matrix: $e_i = Ez_i$. The embedding of all items can be stacked as: $I = [e_1, e_2, ..., e_n]$ where n is the total number of items. The embedding matrix E will be optimized along with the model training process after the user latent factor is learned, and details are provided at the end of this section.

Figure 2 shows that the proposed model consists of a time-specific meta-learning module and a time-evolving recurrent neural network module to generate time-specific user latent factors u^t_{ts} and time-evolving latent factors u^t_{te}, both of which contribute to the final prediction $f^t(u, i)$. Details of them are described in following sections. After both modules are trained, the model learns time-specific user factors u^t_{ts} and time-evolving user factors u^t_{te}. These user factors are merged to interact with the item embedding to provide recommendations for the user. The recommendation for a user u at the current period t is denoted as a vector $\hat{r}^t_u$.

Making recommendations can be viewed as a regression problem. By using the mean square error (MSE) function, the loss for a specific user u is formulated as:

$$\begin{aligned} \mathcal{L}_{\mathcal{T}^t_u}[f^t_{\theta^t_u,\omega}] &= \sum_i ||f^t_{\theta^t_u,\omega}(i) - r^t_{(u,i)}||^2_2, \\ f^t_{\theta^t_u,\omega}(i) &= (u^t_{ts} + u^{t-1}_{te}) \cdot e_i \end{aligned} \tag{3}$$

where $r^t_{(u,i)}$ is user-item interaction (rating or count). The user representation is the vector sum of time-specific u^t_{ts} and time-evolving user factors u^{t-1}_{te} (as a compact single representation reducing the number of trainable parameters that helps to avoid overfitting), and the prediction is achieved by the dot product of u and item embedding i. Note that prediction for the current time includes time-specific user factors from current time period i.e. u^t_{ts} and time-evolving user factors from the previous time period i.e. u^{t-1}_{te}. In general, dynamic recommender systems utilize the information from the previous period to predict for the next period, and we follow this standard setting.

The total loss is formed by aggregating all users in the meta-train set, regularized by the L_2 norm of key model parameters. Let θ^t_u and θ^t denote the local (i.e., user-specific) and global parameters of the time-specific meta-learning module, ω denote the parameters of the time-evolving recurrent neural network module. Training of a dynamic recommendation model can be formulated as:

$$\begin{aligned} \arg\min_{\theta^t,\omega} \sum_{\mathcal{T}^t_u \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}^t_u}[f^t_{\theta^t_u,\omega}] + \frac{\lambda}{2}(||\theta^t||^2_2 + ||\omega||^2_2), \\ \theta^t_u = \theta^t - \alpha \nabla_{\theta^t} \mathcal{L}_{\mathcal{T}^t_u}[f^t_{\theta^t,\omega}] \end{aligned} \tag{4}$$

where θ^t_u is one gradient step from global parameter θ^t of the meta-learned time-specific module with α being the step size and λ is the regularization parameter.

Time-Specific Meta-Learning Module

This module aims to capture time-specific user factors by only considering the information from the current recommendation period. The meta-learner takes input from the specific period, which is a different setting than the existing meta-learning-based recommender systems [17, 5]. In this way, the model can capture the latent factors associated with that specific period to provide more accurate and timely recommendations. We consider each user as a learning task. Our goal is to learn a meta parameter θ^t that represents a time-specific global user representation given the meta-training set. We follow the standard setting of few-shot learning [7], where the distribution over tasks is represented as $p(\mathcal{T})$. The model is trained iteratively by sampling tasks from $p(\mathcal{T})$. The meta-learning module generates time-specific user latent factors (u^t_{ts}) as:

$$u^t_{ts} = f^t_{meta}(\mathcal{T}^t_u; \theta^t) \tag{5}$$

where $\mathcal{T}^t_u$ represents task of a user u at period t. The task $\mathcal{T}^t_u$ includes $\mathcal{S}^t_u$ and $\mathcal{Q}^t_u$. We first pass $\mathcal{S}^t_u$ into f^t_{meta} to adapt user-specific model parameter θ^t_u from the global user model parameter θ^t and then we provide $\mathcal{Q}^t_u$ into the f^t_{meta} to generate time-specific user factors (u^t_{ts}).

We apply an optimization-based meta-learning approach [7] to learn time-specific user factors, as shown in Figure 2. The meta-learning network consists of one input layer, two fully connected hidden layers, and one output layer. The first and second hidden layers have 128 and 64 hidden units with ReLU activation, while the last layer estimates time-specific user factors with a linear function followed by sigmoid activation. The input to the meta-learning model is the item embedding for the users on a particular period. Algorithm 1

shows the training process that learns the model parameters. For the time-specific module, the local update (line 7) is done for the user specific parameter, which is achieved by one or more gradients from the global parameter:

$$\theta_u^t = \theta^t - \alpha \nabla_{\theta^t} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta^t,\omega}] \tag{6}$$

In this update, the loss function is computed with the support set. Similarly, the global update (line 11) is conducted with the new item interactions of each user from the query set for the meta update:

$$\theta^t = \theta^t - \beta \nabla_{\theta^t} \sum_{\mathcal{T}_u^t \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega}] \tag{7}$$

This process continues to find a good global parameter shared by all users in each period.

Time-Evolving Module

User preferences usually change dynamically over time. By capturing the time-evolving factors and integrating them with the time-specific factor, the proposed model can recover the user's true preference more accurately. To this end, we formulate time-evolving user factors (u^t_{te}) for each user using a nested recurrent neural network (RNN):

$$u^t_{te} = f^t_{rnn}(u^{t-1}_{te}, D^t_u; \omega) \tag{8}$$

where D^t_u is the set of items that user u interacted with at time t, u^{t-1}_{te} is the previous time period time-evolving user factors, and ω is the network parameter. Notice that the input and output of the RNN are both latent variables instead of observations. We use SGD to update the parameter of RNN:

$$\omega = \omega - \gamma \nabla_\omega (\mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega}] + \frac{\lambda}{2}||\omega||_2^2) \tag{9}$$

where γ is the step size. As shown in the time-evolving module of Figure 2, the vector representation of a hidden layer u^t_{te} is a time-evolving factor of user u at period t and helps to propagate influence from the previous period to the next period [36]. The updates of time-specific user factors through meta-learning and time-evolving user factors through nested RNN are summarized in Algorithm 1. The recommendation process is summarized in Algorithm 2.

Joint Item Embedding Optimization. Let $\mathcal{L}_{emb}$ denote a differentiable loss function used to train the embedding matrix E. And let $\mathcal{G}$ denote the decoding module, which is followed by attribute-wise sigmoid transformation:

$$\begin{aligned} d_i &= \mathcal{G}(Ez_i) \\ [\hat{z}_i]_j &= \text{sigmoid}(\eta_j^T d_i) = \frac{1}{1+\exp(-\eta_j^T d_i)} \quad \forall j \in \{1,...,K\} \end{aligned} \tag{10}$$

where z_i is the original item representation in a binary vector, K is the length of z_i, E is the embedding matrix, d_i is the decoded item representation, η is the parameter for attribute-wise Sigmoid transformation, and $\hat{z}_i$ is the recovered item representation. The loss function for learning the embedding matrix is a negative log-likelihood and is represented as:

$$\mathcal{L}_{emb} = -\sum_{i \in I} \sum_j [z_i]_j \log [\hat{z}_i]_j \tag{11}$$

Algorithm 1: Model training

Require: Set of time periods: $\mathcal{H}$
Require: Hyperparameters: α, β, γ
1: **for** $t \in \mathcal{H}$ **do**
2: Initialize meta learner, θ^t
3: **while** not converge **do**
4: Sample tasks $\mathcal{T}_u^t \sim p(\mathcal{T})$
5: **for** all $\mathcal{T}_u^t$ **do**
6: Sample support set, $\mathcal{S}_u^t \in \mathcal{T}_u^t$ for the local update
7: Perform local update with $\mathcal{S}_u^t$ for time-specific module using Equation (6)
8: Sample query set $\mathcal{Q}_u^t \in \mathcal{T}_u^t$ for the meta update
9: Update time-evolving module with $\mathcal{D}_u^{t-1}$ using Equation (9)
10: **end for**
11: Perform meta update with $\mathcal{Q}_u^t$ for time-specific module using Equation (7)
12: **end while**
13: **end for**

Algorithm 2: Recommendation for time-specific cold-start users

Require: Trained meta parameter θ^t, RNN parameter ω, recommendation time period t
1: Identify cold-start user set for t
2: **for** each user u in the set **do**
3: Form support set $\mathcal{S}_u^t$ from current interactions
4: Perform local update with $\mathcal{S}_u^t$ for time-specific module using Equation (6)
5: Compute user factors using Equations (5) and (8)
6: Make recommendation using Equation (3)
7: **end for**

Other designs for item embedding with a differentiable loss function can also be applied.

To jointly train the embedding matrix, time-specific and time-evolving modules, we combine those losses, and optimize the total loss with respect to θ^t, ω and E.

$$\begin{aligned} &\arg\min_{\theta^t,\omega,E} \sum_{\mathcal{T}_u^t \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega,E}] + \xi \mathcal{L}_{emb} + \frac{\lambda}{2}(||\theta^t||_2^2 + ||\omega||_2^2) \\ &\theta_u^t = \theta^t - \alpha \nabla_{\theta^t} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta^t,\omega,E}] \end{aligned} \tag{12}$$

where ξ is the weight to be tuned. If ξ is set to a very large value, matrix E is determined only by $\mathcal{L}_{emb}$.

Note that the encoded item embedding is used for both modules. By fixing θ_u^t and ω, it is possible to back-propagate and calculate the gradient with respect to E, which is updated in each task as

$$E = E - \gamma \nabla_E (\mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega,E}] + \mathcal{L}_{emb}) \tag{13}$$

Also notice that $\mathcal{L}_{emb}$ is not dependent on θ^t and ω. Therefore, when embedding matrix is fixed, the loss function reduces to Eq (4), and θ^t and ω are updated without considering $\mathcal{L}_{emb}$.

Experiments

We conduct experiments on two movie datasets: *Netflix* and *MovieLens-1M* that consist of users' ratings of movies as explicit feedback and one music dataset: *Last.fm* that consists of users' play counts of music tracks as implicit feedback. Besides reporting the overall recommendation performance and comparing with state-of-the-art baselines, we also investigate key properties of the model, including: (1) each module's performance when used in isolation, (2) impact of varying time period lengths, (3) recommendation performance with no interactions in the current period, and (4) impact of hyper-parameters in the model. Experimental settings, datasets details, and analysis of the above key properties are presented in the Appendix [20].

Methods for Comparison. For comparison, we include matrix factorization based static and dynamic models, deep learning based models, graph models, sequential models, and meta-learning models:

- *Matrix factorization (MF):* The standard MF model SVD++ [14] that also exploits both explicit and implicit feedback is used here as a static baseline.
- *Dynamic models:* We use timeSVD++ [15], collaborative Kalman Filter (CKF) [10], and dynamic Poisson factorization (DPF) [3] as the time-evolving models.
- *Deep learning models:* We use Wide and deep [4], DeepFM [11] as static, and DIEN [37] as a dynamic models for deep learning-based recommendation. However, most of them are developed for click-through rate prediction in their original forms.
- *Graph-based model:* Most graph-based models are designed for static settings. For comparison, we use graph convolutional matrix completion (GC-MC) [2], which models recommendation as link prediction in the graph, and neural graph collaborative filtering (NGCF) [31] that utilizes embedding propagation over user-item graphs.
- *Sequential model:* We use Sequential Recommendation via Convolutional Sequence Embedding (Caser) [26], which models recommendation as a unified and flexible structure to capture both preferences and sequential patterns, and transformer-based sequential recommendation model (SASRec) [13] in our comparison.
- *Meta-learning models:* We follow the model-agnostic meta-learning model (MAML) to implement the meta-learning model similar to MeLU [17]. We also compared with the meta-learning model in [28] that focuses on item cold-start problem (referred to as ML-ICS). The model is also designed for the classification setting, so we have to make adjustments to fit into our context.

Evaluation Metrics. For evaluation, we analyze the experimental results in terms of both the deviation of predicted values from the ground truth and the errors of the ranking sequences. We use Root Mean Squared Error (RMSE) and Normalized Discounted Cumulative Gain (NDCG) averaged across all test users. RMSE is usually reported for explicit data, while NDCG is usually reported for implicit data:

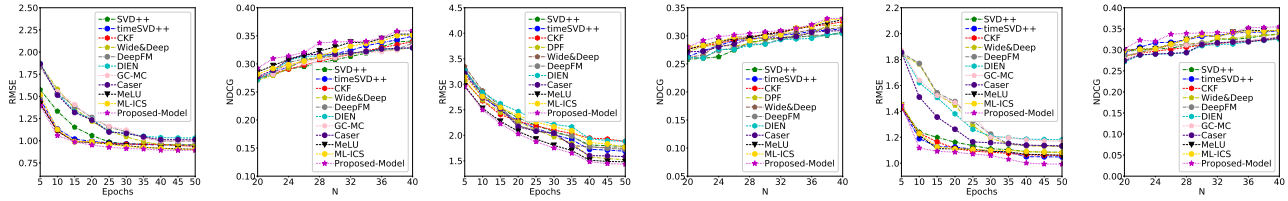
$$\begin{aligned} \text{RMSE} &= \sqrt{\sum_{r_{u,i} \in O} (\hat{r}_{u,i} - r_{u,i})^2 / |O|}, \\ \text{NDCG}_u &= \sum_n \frac{rel_n^{pred}}{\log_2(1+n)} / \sum_n \frac{rel_n^{ideal}}{\log_2(1+n)} \end{aligned} \tag{14}$$

where O is the observation set for the test set and rel_n is the relevancy of n^{th} item in the ranking sequence for user u, which is binary for implicit data or the rating for explicit data. To penalize the negative feedback, we linearly mapped the ratings to a range of [-1,1]. The NDCG is the fraction of Discounted Cumulative Gain (DCG) of recommendation result over the ideal DCG.

Recommendation Performance

The experimental results are shown in Figures 3a-3c. We evaluate NDCG based on the top N recommendation list and RMSE based on the training epochs. The RMSE is stable after 30 and 40 epochs in both movies and music datasets, respectively. The average results of NDCG and RMSE considering all periods with the range of deviation are shown in Table 2, for the Netflix, Last.fm, and MovieLens datasets, respectively. The proposed model clearly demonstrates the advantage of combining time-specific and time-evolving user factors that lead to a superior recommendation accuracy as compared with other competitive models. Both explicit and implicit datasets are highly sparse, and MF models' performance is poor due to the sparse interactions. Also, MF models suffer from cold-start problems, and thus their performances are fairly limited, as shown in Table 2. Similarly, deep learning models require sufficient training data and hence largely suffer in the few-shot recommendation setting. Moreover, these models might need extra side information, like user profile and item details, for better recommendations. For example, DIEN needs cleverly chosen interest features like user behavior, and the absence of those features limits its performance, as shown in Table 2. Similarly, the poor performance of graph-based models in both movie datasets implies that these methods are insufficient to handle cold-start problems. Like other existing models, the performance of a sequential model is less effective for the cold-start users in all three datasets. The reason could be that the model is less effective in capturing long-term user preferences. In contrast, the meta-learning approaches show better results by leveraging shared knowledge across the users. However, in the time-specific cold-start setting, test users have very limited interactions. In particular, the meta-learning model doesn't benefit from time-evolving aspects of the user interests, and thus underperforms the proposed model.

We use an illustrative example to further demonstrate how the proposed model effectively captures the underlying user interest and its evolution in Figure 4. The recommended movie genres are compared to the user's favorite genres based on the provided true ratings. The result shows that the recommendation matches user's changing taste over time well. It is also interesting to see that the proposed model

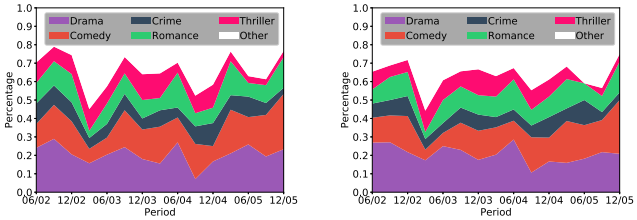


(a) Netflix (b) Last.fm (c) MovieLens-1M

Figure 3: NDCG based on the top N recommendations and RMSE based on the training epochs

Table 2: Recommendation Results (RMSE and NDCG)

Category	Model	Netflix		Last.fm		MovieLens-1M	
		RMSE	NDCG	RMSE	NDCG	RMSE	NDCG
MF	SVD++	0.9797±0.03	0.2915	1.7829±0.08	0.2882	1.0825±0.04	0.3023
Dynamic	timeSVD++	0.9538±0.06	0.3115	1.6912±0.11	0.2962	1.0483±0.03	0.3224
	CKF	0.9337±0.04	0.3130	1.5316±0.32	0.3018	1.0652±0.04	0.3151
	DPF	N/A	N/A	1.5227±0.43	0.3085	N/A	N/A
Deep Learning	Wide and Deep	0.9904±0.04	0.2864	1.7253±0.22	0.2727	1.1364±0.06	0.2932
	DeepFM	0.9811±0.03	0.2930	1.6815±0.21	0.2971	1.1723±0.05	0.2882
	DIEN	1.0345±0.04	0.2832	1.9225±0.26	0.2714	1.1872±0.14	0.2843
Graph	GC-MC	1.0760±0.03	0.2901	N/A	N/A	1.1704±0.08	0.2913
	NGCF	1.0321±0.03	0.3026	1.5612±0.23	0.2896	1.1216±0.05	0.3103
Sequential	Caser	1.0124±0.03	0.3101	1.5824±0.31	0.2931	1.1339±0.08	0.3012
	SASRec	N/A	0.3246	N/A	0.3103	N/A	0.3238
Meta-Learning	MeLU	0.9213±0.05	0.3232	1.2580±0.28	0.3122	1.0685±0.08	0.3214
	ML-ICS	0.9332±0.04	0.3173	1.2408±0.24	0.3142	1.0845±0.06	0.3244
Proposed	**Ours**	**0.8925±0.03**	**0.3472**	**1.2203±0.16**	**0.3385**	**0.9945±0.08**	**0.3351**



(a) Ground Truth (b) Recommended

Figure 4: Dynamic trend of movie genres in Netflix: ground truth (a) vs model recommended (b)

Table 3: Time-specific popular movies learned and predicted by the meta-learning module

Users	Movies
Training users	['Best in Show', 'Chicken Run', 'Sommersby', 'Bedrooms and Hallways', 'The Mod Squad']
Test user (ID:5636)	['Mr. Mom', 'Best in Show', 'Shower', "We're No Angels", 'Groundhog Day']
Test user (ID:5539)	['Best in Show', 'An Ideal Husband', 'Life Is Beautiful', 'Breaking Away', 'Kramer vs. Kramer']

accurately detects some dramatic changes in user's ratings (e.g., from 12/02 to 06/03 and from 06/04 to 12/04), which were likely to be caused by some time-specific factors.

We further present an example to show how the meta-learning module effectively captures time-specific factors in the form of popular trends in a specific period from the global user space and transfers the (meta) knowledge to the cold-start users with very limited interactions. Table 3 demonstrates top-5 time-specific (period 4) popular movies learned by the meta-learning module, which shares that knowledge with the test users (i.e., two users are shown in Table 3 and some time-specific popular movies like 'Best in Show' are recommended to them). This example demonstrates how our model provides effective recommendations to users by capturing time-specific factors.

Conclusion

In this paper, we formulate a novel time-sensitive cold-start problem and present a dynamic recommendation framework to address its unique challenges. The framework integrates a time-sensitive meta-learning module with a time-evolving recurrent module. The former handles the user cold-start problem by learning global knowledge among users from their interaction information in the current recommendation period. This module is jointly optimized with the time-evolving recurrent module that captures a user's gradually shifted preferences. A merged user representation is generated using the two modules' outputs and interacts with the item embedding to provide the final recommendations. Experimental results of real-world dynamic datasets and comparison with the state-of-the-art models clearly demonstrate the performance advantage of the proposed model.

Acknowledgement

This research was supported in part by an NSF IIS award IIS-1814450, an ONR award N00014-18-1-2875, and by the Army Research Office under grant number W911NF-21-1-0236. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agency.

Ethical Impact

The proposed model is generally applicable to many real-world applications where maintaining existing users' participation is critical, such as product recommendation in e-business and content recommendation in social media. Besides, this model can operate on multiple platforms such as health, education, and e-commerce to handle those sensitive users in the system. Since the proposed model does not rely on using users' profile information, it has the potential to improve the privacy protection of recommender systems.

References

[1] Bengio, S.; Bengio, Y.; Cloutier, J.; and Gecsei, J. 1992. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, volume 2. Univ. of Texas.

[2] Berg, R. v. d.; Kipf, T. N.; and Welling, M. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

[3] Charlin, L.; Ranganath, R.; McInerney, J.; and Blei, D. M. 2015. Dynamic poisson factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, 155–162.

[4] Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 7–10.

[5] Du, Z.; Wang, X.; Yang, H.; Zhou, J.; and Tang, J. 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.

[6] Fang, H.; Zhang, D.; Shu, Y.; and Guo, G. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)*, 39(1): 1–42.

[7] Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1126–1135. JMLR. org.

[8] Funk, S. 2006. Incremental SVD method.

[9] Goldberg, D.; Nichols, D.; Oki, B. M.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12): 61–70.

[10] Gultekin, S.; and Paisley, J. 2014. A collaborative kalman filter for time-evolving dyadic processes. In *2014 IEEE International Conference on Data Mining*, 140–149. IEEE.

[11] Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247*.

[12] Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; and He, Q. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*.

[13] Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, 197–206. IEEE.

[14] Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 426–434.

[15] Koren, Y. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 447–456.

[16] Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37.

[17] Lee, H.; Im, J.; Jang, S.; Cho, H.; and Chung, S. 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.

[18] Lu, Y.; Fang, Y.; and Shi, C. 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.

[19] Mnih, A.; and Salakhutdinov, R. R. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*, 1257–1264.

[20] Neupane, K.; Zheng, E.; Kong, Y.; and Yu, Q. 2021. Appendix: A Dynamic Meta-Learning Model for Time-Sensitive Cold-Start Recommendations. https://github.com/ritmininglab/A-Dynamic-Meta-Learning-Model-for-Time-Sensitive-Cold-Start-Recommendations/blob/main/Appendix.pdf.

[21] Pan, F.; Li, S.; Ao, X.; Tang, P.; and He, Q. 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.

[22] Sahoo, N.; Singh, P. V.; and Mukhopadhyay, T. 2012. A hidden Markov model for collaborative filtering. *Mis Quarterly*, 1329–1356.

[23] Schmidhuber, J. 1987. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Ph.D. thesis, Technische Universität München.

[24] Su, X.; and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009.

[25] Sun, J. Z.; Parthasarathy, D.; and Varshney, K. R. 2014. Collaborative kalman filtering for dynamic matrix factorization. *IEEE Transactions on Signal Processing*, 62(14): 3499–3509.

[26] Tang, J.; and Wang, K. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 565–573.

[27] Uyangoda, L.; Ahangama, S.; and Ranasinghe, T. 2018. User profile feature-based approach to address the cold start problem in collaborative filtering for personalized movie recommendation. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, 24–28. IEEE.

[28] Vartak, M.; Thiagarajan, A.; Miranda, C.; Bratman, J.; and Larochelle, H. 2017. A meta-learning perspective on cold-start recommendations for items. In *Advances in neural information processing systems*, 6904–6914.

[29] Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, 3630–3638.

[30] Wang, S.; Hu, L.; Wang, Y.; Cao, L.; Sheng, Q. Z.; and Orgun, M. 2019. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*.

[31] Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 165–174.

[32] Wei, T.; Wu, Z.; Li, R.; Hu, Z.; Feng, F.; He, X.; Sun, Y.; and Wang, W. 2020. Fast Adaptation for Cold-start Collaborative Filtering with Meta-learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, 661–670. IEEE.

[33] Xie, F.; Chen, L.; Ye, Y.; Zheng, Z.; and Lin, X. 2018. Factorization machine based service recommendation on heterogeneous information networks. In *2018 IEEE International Conference on Web Services (ICWS)*, 115–122. IEEE.

[34] Xie, R.; Wang, Y.; Wang, R.; Lu, Y.; Zou, Y.; Xia, F.; and Lin, L. 2021. Long Short-Term Temporal Meta-learning in Online Recommendation. *arXiv preprint arXiv:2105.03686*.

[35] Yao, Y.; Tong, H.; Yan, G.; Xu, F.; Zhang, X.; Szymanski, B. K.; and Lu, J. 2014. Dual-regularized one-class collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 759–768.

[36] Zhang, Y.; Dai, H.; Xu, C.; Feng, J.; Wang, T.; Bian, J.; Wang, B.; and Liu, T.-Y. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[37] Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; and Gai, K. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5941–5948.