HyperBandit: Contextual Bandit with Hypernewtork for Time-Varying User Preferences in Streaming Recommendation

Chenglei Shen
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
chengleishen9@ruc.edu.cn

Xiao Zhang*
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
zhangx89@ruc.edu.cn

Wei Wei
CCIIP Laboratory
Huazhong University of Science and Technology
Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL)
Wuhan, China
weiw@hust.edu.cn

Jun Xu
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
junxu@ruc.edu.cn

ABSTRACT

In real-world streaming recommender systems, user preferences often dynamically change over time (e.g., a user may have different preferences during weekdays and weekends). Existing bandit-based streaming recommendation models only consider time as a timestamp, without explicitly modeling the relationship between time variables and time-varying user preferences. This leads to recommendation models that cannot quickly adapt to dynamic scenarios. To address this issue, we propose a contextual bandit approach using hypernetwork, called HyperBandit, which takes time features as input and dynamically adjusts the recommendation model for time-varying user preferences. Specifically, HyperBandit maintains a neural network capable of generating the parameters for estimating time-varying rewards, taking into account the correlation between time features and user preferences. Using the estimated time-varying rewards, a bandit policy is employed to make online recommendations by learning the latent item contexts. To meet the real-time requirements in streaming recommendation scenarios, we have verified the existence of a low-rank structure in the parameter matrix and utilize low-rank factorization for efficient training. Theoretically, we demonstrate a sublinear regret upper bound against the best policy. Extensive experiments on real-world datasets show that the proposed HyperBandit consistently outperforms the state-of-the-art baselines in terms of accumulated rewards.

CCS CONCEPTS

• **Theory of computation → Online learning theory**; • **Information systems → Recommender systems**.

*Xiao Zhang is the corresponding author. The work was partially done at Beijing KeyLaboratory of Big Data Management and Analysis Methods.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614921

KEYWORDS

contextual bandit; hypernetwork; streaming recommendation; time-varying user preference

ACM Reference Format:
Chenglei Shen, Xiao Zhang, Wei Wei, and Jun Xu. 2023. HyperBandit: Contextual Bandit with Hypernewtork for Time-Varying User Preferences in Streaming Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614921

1 INTRODUCTION

While the demand for personalized recommendations has increased due to the growth of online platforms and user-generated content, it is crucial to emphasize that the recommendation models need to be updated frequently and integrated with online recommender systems to ensure optimal performance in real-time. This makes streaming recommendation a highly active area of research aimed at continuously updating the model based on users' latest interactions with the platform and delivering relevant and timely suggestions to users [6, 7, 19, 34, 35, 40].

Nonetheless, streaming recommendation confronts a significant challenge in the form of the phenomenon of time-varying user preferences [9]. Users' preferences change dynamically over time due to several factors such as seasonality, holidays, or circadian rhythm. As illustrated in Fig. 1, users tend to check in at places such as "Office" and "Coffee Shop" on weekday mornings, while at places like "Gym / Fitness Center" and "Church" on weekend mornings, demonstrating a weekly periodicity. In contrast to morning preferences, users tend to visit bars and spend time at home during evening hours regardless of whether it is a weekday or weekend, indicating a daily periodicity. Another example of short video recommendation is that users exhibit a tendency to watch cartoons specifically on weekends, while preferring other types of content on weekdays. These recurring patterns highlight the importance of considering time-varying user preferences to avoid sub-optimal recommendations. Consequently, devising effective approaches to address the issue of users' periodic time-varying preference is critical for achieving high-quality streaming recommendation.

As a classic framework for online learning, multi-armed bandit (MAB) algorithms have gained significant attention in recent years. A variation of MAB, known as contextual bandits [21, 22, 36],

Figure 1: The illustrations of the periodic shift on points-of-interest (POI) dataset Foursquare-NYC, representing word clouds of POI for morning/night on weekdays/weekends.

has achieved considerable success in various online services by utilizing both user feedback and contextual information related to users and items, which make it particularly advantageous in streaming recommendation. Most existing contextual bandit algorithms are constructed under stationary environment, i.e. users' preferences remain static over time [1, 15, 22]. However, the environment is always non-stationary in reality indicating time-varying user preferences. Some studies have relaxed the assumption to the piecewise stationary environment [36, 37], which enables algorithms to adaptively detect user preferences change points and discard learned model parameters for relearning. These methods can lead to performance fluctuations, particularly when dealing with periodic changes in user preferences. The main issue is their inability to capture the periodic nature of user preferences in an online fashion, often triggering model retraining even for previously encountered periods. Currently, there's a significant research gap in the field of streaming recommendations within periodic environments.

In this paper, we focus on a realistic environment setting where the reward function (i.e., the generation mechanism of user feedback) exhibits periodicity over time. Specifically, a large time period can be divided into multiple smaller periods in a periodic manner (e.g., based on the specific day of the week and different time slots within a day), and the reward function demonstrates a similar distribution whenever the same time period is encountered. Moreover, these time periods can be observed by the model and utilized for periodicity modeling and online adjustment of its user preference module in various streaming recommendation scenarios.

As a specific solution to the aforementioned process, we propose a novel contextual bandit algorithm called HyperBandit, which consists of two levels of model structures: 1). A *bandit policy* is designed to learn the latent features of items in an online fashion and combine them with the user preference matrix to execute online recommendations with effective exploration. 2). A *hypernetwork* takes the information of the time period as inputs and generates the parameters of the user preference matrix in the bandit policy. This hypernetwork captures the periodicity of user preferences over time and enables efficient online updating through low-rank factorization. Through extensive experiments on streaming recommendation tasks, such as short video and POI recommendations, we showcase the efficiency and effectiveness of HyperBandit.

2 RELATED WORK

Hypernetworks (HNs) have been introduced by Ha et al. [14], drawing inspiration from the genotype-phenotype relation in cellular biology. HNs present an approach of using one network (hypernetwork) to generate weights for a second network (target network). In recent years, HNs are widely used in various domains such as computer vision [20], language modeling [30], sequence decoding [24], continual learning [31], federated learning [28], multi-objective optimization [8, 25], and hyperparameter optimization [23]. Navon et al. [25] proposed a unified model to learn the Pareto front based on HNs that can be applied to a specific objective preference at inference time. von Oswald et al. [31] presented a task-aware method for continual reinforcement learning using HNs, which allows the entire network to change between tasks as well as retaining performance on previous tasks. HNs have been widely applied in offline learning, but there is a lack of research on how to enhance the controllability of models through hypernetworks in online learning and streaming applications.

Bandits in non-stationary environment have attracted extensive attention in both theory and applications in recent years. One common setting for non-stationary environments is the abruptly changing or piecewise-stationary environment, where the environment undergoes sudden changes at unknown time points while remaining stationary between consecutive change points. Under the piecewise-stationary assumption, the problem has been well studied in the classical context-free setting [13, 17, 29, 39]. Yu et al. [39] proposed a windowed mean-shift detection algorithm to identify potential abrupt changes in the environment. They provided an upper bound on regret of $O(\Gamma_T \log(T))$ for their algorithm, where Γ_T represents the number of ground-truth changes up to time T. Within the contextual bandit setting, limited attention has been given to addressing non-stationary environments [16, 36, 37]. Wu et al. [36] developed a hierarchical bandit algorithm capable of detecting and adapting to changes by maintaining multiple contextual bandits. More recently, Xu et al. [37] addressed the challenge of time-varying preferences by employing a change-detection procedure to identify potential changes on the preference vectors. However, little attention has been given to addressing the issue of periodic reward drift that this paper focuses on.

3 PROBLEM FORMULATION

3.1 Bandit-based Streaming Recommendation

Streaming recommendation can be formulated as a problem of sequential decision making, where the online service platform recommends the most relevant item $a \in \mathcal{A}$ (such as videos, music, or POIs) to a user $u \in \mathcal{U}$ in an online manner. Contextual bandit algorithms are well-suited for addressing streaming recommendation problems. More specifically, the candidate item set $\mathcal{A}$ could be viewed as the action space of the bandit algorithm, while the

context space $\mathcal{S}$ summarizes the feature information of users and items, where each item a and user u can be associated with context feature vectors denoted by $\boldsymbol{c}_a$ and $\boldsymbol{c}_u$, respectively. At time step t, given a subset of the action space $\mathcal{A}_t \subseteq \mathcal{A}$ and a user, an item is selected by a recommendation policy and recommended to the user. After one item is recommended, the item may be clicked by the user (i.e, positive user feedback) or skipped (i.e, negative user feedback). Thereafter the true reward defined on the user feedback is received and could be used for updating the current recommendation policy, which will be adopted for the next recommendation.

The above process can be formalized as a contextual bandit problem for streaming recommendation, and represented using a 4-tuple $\langle \mathcal{A}, \mathcal{S}, \pi, r \rangle$:

Action space $\mathcal{A}$ denotes a given candidate action set, where each action (also called arm) corresponds to a specified candidate item. At each time step, a dynamic action space is selected as the candidate item set for recommendation. That is, at time step t, a candidate item set $\mathcal{A}_t \subseteq \mathcal{A}$ is recalled by some strategy, and choosing an action a_{I_t} from $\mathcal{A}_t$ means that the corresponding item is recommended to the user, where $I_t \in |\mathcal{A}_t|$ denotes the index of the recommended item at time t.

Context space $\mathcal{S}$ summarizes the context feature information of users and items, denoted by $\boldsymbol{c}_u \in \mathbb{R}^{d_u}$ and $\boldsymbol{c}_a \in \mathbb{R}^{d_a}$, respectively. In this paper, in particular, we consider splitting the item context $\boldsymbol{c}_a \in \mathbb{R}^{d_a}$ into two parts: the observed features $\boldsymbol{s}_a \in \mathbb{R}^{o_a}$, and the latent features $\boldsymbol{x}_a \in \mathbb{R}^{l_a}$ that needs to be learned. Here, $\boldsymbol{c}_a = [\boldsymbol{s}_a^\intercal, \boldsymbol{x}_a(t)^\intercal]^\intercal$ and the dimension of $\boldsymbol{c}_a$ is given by $d_a = o_a + l_a$.

Policy $\pi : \mathcal{S} \to \mathcal{A}$ describes the decision-making rule of an agent (i.e., the recommendation model), which selects an action for execution according to the relevance score of each action. at time t, given a candidate item set $\mathcal{A}_t$ and user $u \in \mathcal{U}$, a *relevance score function* f_t treats context features of user and item in context space (i.e., $\boldsymbol{c}_u$ and $\boldsymbol{c}_a$) as inputs and determines which action to take: $a_{I_t} := \arg\max_{a \in \mathcal{A}_t} f_t(\boldsymbol{c}_u, \boldsymbol{c}_a)$.

Reward r is defined upon the user feedback. Specifically, at time t, after recommending the item $a_{I_t} \in \mathcal{A}_t$ to a user u, a corresponding reward $r(u, a_{I_t}) \in \{0, 1\}$ is observed, which implicitly indicates whether the user feedback is negative or positive to the item a_{I_t}. However, the feedbacks from the same user towards the same item at different time may be quite different, which means that time-varying user preferences exist in it.

Table 1 summarizes the notations used throughout the paper.

3.2 Time-Varying User Preferences

In this section, we formally describe the time-varying user preferences mentioned in the introduction.

As defined in Table 1, we first introduce the time period variable p to measure specific temporal patterns, including hours of the day and different days of the week. Specifically, we divide a week into seven days, from Monday to Sunday, and further divide each day into the following five sessions: the morning (8:00 AM to 11:30 AM), the noon (11:30 AM to 2:00 PM), the afternoon (2:00 PM to 5:30 PM), the night (5:30 PM to 10:00 PM), and the remaining period. Then, the time period variable, p, encompasses 35 distinct values spanning from 0 to 34 in a sequential order. Each time period can be encoded to derive its respective *time period embedding*, denoted as $\boldsymbol{s}_p \in \mathbb{R}^{d_p}$.

Table 1: A summary of notations.

Symbol	Explanation
$[n]$	$[n] := [1, 2, \ldots, n]$
t	Time step $t \in [T]$
$\mathcal{A}$	Action space, i.e., the candidate item set
$\lvert\mathcal{A}\rvert$	The cardinality of set $\mathcal{A}$
$\boldsymbol{c}_u \in \mathbb{R}^{d_u}$	Context feature vector of a user u
$\boldsymbol{c}_a \in \mathbb{R}^{d_a}$	Context feature vector of a candidate item a
$\boldsymbol{s}_a \in \mathbb{R}^{o_a}$	Observed features of a candidate item a
$\boldsymbol{x}_a \in \mathbb{R}^{l_a}$	Latent features of a candidate item a
$p \in \mathcal{P}$	Time period variable, takes values in the range $\mathcal{P} := \{0, 1, \ldots, 34\}$, representing the 35 time periods within a week
$\boldsymbol{s}_p \in \mathbb{R}^{d_p}$	Time period embedding of time period p
$\boldsymbol{\Theta}_p^*$	True user preference matrix at the time period p

Under the traditional assumption of a stationary environment, the mechanism of user feedback should be consistent at every time period. That is, the *reward generation probability*, represented as $\Pr\{r(u, a) = 1 \mid \boldsymbol{c}_u, \boldsymbol{c}_a\}$, is assumed to remain constant across all time step $t \in [T]$. This implies that the level of preference that user u has for the recommended item a is independent of the specific time at which the recommendation is made. However, in real-world streaming recommender systems, users' preferences change with time periodically, which has been observed in [12]. For example, users usually visit office at weekday morning and bars at night. That is, the user feedback towards office may be different at different time period. In other words, given the context $\boldsymbol{c}_u, \boldsymbol{c}_a$, the current time period p and the corresponding time period embedding $\boldsymbol{s}_p$, the following inequality may hold:

$$\Pr\left\{r(u, a) = 1 \mid \boldsymbol{c}_u, \boldsymbol{c}_a, \boldsymbol{s}_{p=i}\right\} \neq \Pr\left\{r(u, a) = 1 \mid \boldsymbol{c}_u, \boldsymbol{c}_a, \boldsymbol{s}_{p=j}\right\}, \quad (1)$$

where $i \neq j$ and $i, j \in \{0, \ldots, 34\}$, and $\Pr\left\{r = 1 \mid \boldsymbol{c}_u, \boldsymbol{c}_a, \boldsymbol{s}_p\right\}$ denotes the *time-varying reward generation probability* indicating how much the user u prefers the recommended item a at time period p.

Formally, we can represent the observed reward generated by the time-varying reward generation probability as $r(u, a, p)$. Given a user u, a item a, and a time period p, the generation process of the observed reward can be formalized as $r(u, a, p) := r^*(u, a, p) + \eta$, where $r^*(u, a, p)$ represents the *true reward*, and η is a random variable drawn from a distribution with zero mean. This additional error term η captures the noise or uncertainty present in the observations. Clearly, we have the following expected reward:

$$\mathbb{E}[r(u, a, p)] = r^*(u, a, p) = \Pr\left\{r(u, a, p) = 1 \mid \boldsymbol{c}_u, \boldsymbol{c}_a, \boldsymbol{s}_p\right\}.$$

Next, we make specific assumptions about the form of the expected reward $\mathbb{E}[r(u, a, p)]$, i.e., the true reward. One straightforward approach is to concatenate the time period embedding with the context feature vectors. However, existing research [10] has shown that directly concatenating features from different spaces can make it difficult to capture meaningful information (i.e., time-varying information in this paper). To address this issue, we extend the existing linear expected reward in the contextual bandit setting by introducing the true user preference matrix $\boldsymbol{\Theta}_p^*$. Then, we can specify the true reward as the following *time-varying true reward*:

$$r^*(u, a, p) := \boldsymbol{c}_a^\intercal \boldsymbol{\Theta}_p^* \boldsymbol{c}_u, \quad (2)$$

where the *true user preference matrix* $\Theta_p^* \in \mathbb{R}^{d_a \times d_u}$ is utilized to map the user contexts through a linear mapping, taking into account the time period p as well as its embedding s_p as conditions. By applying this mapping, the resulting vector $\Theta_p^* c_u$ can effectively capture the time-varying user preferences across different time periods. In particular, when $d_a = d_u$ and Θ_p^* is the identity matrix, the time-varying true reward degenerates to a fixed true reward in the traditional contextual bandit setting.

4 HYPERBANDIT: THE PROPOSED ALGORITHM

4.1 Algorithm Overview

Fig. 2 illustrates the structure of HyperBandit. Given the current time period as input, a hypernetwork generates a user preference matrix that maps user context features to a time-aware preference space. The mapped features, along with the item context features, are then utilized by the bandit policy to recommend a suitable item to the current user.

HyperBandit consists of the following two components. Firstly, it utilizes a bandit policy to update the latent features of items at each time step, enabling the preservation of time-varying latent features to capture distribution shifting. Secondly, it employs a hypernetwork that is trained in a mini-batch manner to adaptively adjust the user preference matrix in the bandit policy for a given time period. The algorithm's detailed procedure is outlined in Algorithm 1. It's important to note that the user preference matrix is estimated using two low-rank matrices during training. This specific training technique will be discussed in detail in Sec. 4.3.

4.2 Hypernetwork Assisted Bandit Policy

4.2.1 Bandit Policy using User Preference Matrix. To estimate the time-varying true reward and account for the user preference shift in each time period, we propose a novel bandit policy that utilizes an estimate of the true user preference matrix Θ_p^* in Eq. (2). Specifically, the *estimated user preference matrix*, denoted as $\Theta_p \in \mathbb{R}^{d_a \times d_u}$, captures the changes in user preferences during time period p. We estimate Θ_p using a hypernetwork, which will be introduced in Sec. 4.2.2. The estimated user preference matrix allows us to adapt our bandit policy to the evolving user preferences. Formally, assuming that time step t belongs to time period p, given a user context $c_u \in \mathbb{R}^{d_u}$ and the estimated user preference matrix, the following ridge regression over the current interaction history is employed to estimate the item context $c_a(t)$ at time $t \in [T]$:

$$c_a(t) = \arg\min_{c_a \in \mathbb{R}^{d_a}} \sum_{(u,a,r) \in \mathcal{H}_t} \left[c_a^\intercal \Theta_p c_u - r(u,a,p) \right]^2 + \lambda \|c_a\|_2^2, \quad (3)$$

where $\hat{r}_{u,a,p} := c_a^\intercal \Theta_p c_u$ denotes the *estimated time-varying reward*, $\mathcal{H}_t := \{(u_k, a_{I_k}, r_k)\}_{k \in [t]}$ represents the *interaction history* up to time t, (u_k, a_{I_k}, r_k) denotes that the policy recommended item a_{I_k} to user u_k at time k and received a reward r_k, and $\lambda > 0$ is the regularization parameter.

To reduce the uncertainty of user preference estimations, we introduce the observed item features. Specifically, we split the context $c_a(t)$ at time t of item $a \in \mathcal{A}_t$ into two parts, represented as $c_a(t) := [s_a^\intercal, x_a(t)^\intercal]^\intercal \in \mathbb{R}^{d_a}$, which includes: the observed features $s_a \in \mathbb{R}^{o_a}$, and the latent features $x_a(t) \in \mathbb{R}^{l_a}$ that needs to

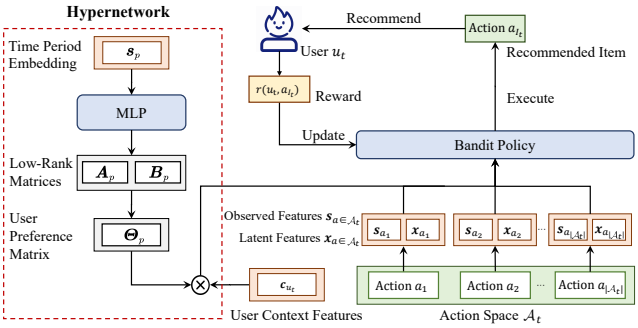


Figure 2: The structure of HyperBandit at time t.

be learned online, where $d_a = o_a + l_a$. Accordingly, we redefine the estimated user preference matrix as $\Theta_p = \left[\Theta_p^{s\intercal}, \Theta_p^{x\intercal}\right]^\intercal$, where $\Theta_p^s \in \mathbb{R}^{o_a \times d_u}$ corresponds to the observed item features s_a, and $\Theta_p^x \in \mathbb{R}^{l_a \times d_u}$ corresponds to the latent item features $x_a(t)$. As a result, we can rewrite the ridge regression in Eq. (3) as follows:

$$\begin{aligned} & x_a(t) \quad (4) \\ &= \arg\min_{x_a \in \mathbb{R}^{l_a}} \sum_{(u,a,r) \in \mathcal{H}_t} \left\{ [s_a^\intercal, x_a(t)^\intercal] \Theta_p c_u - r(u,a,p) \right\}^2 + \lambda \|x_a\|_2^2 \\ &= \arg\min_{x_a \in \mathbb{R}^{l_a}} \sum_{(u,a,r) \in \mathcal{H}_t} \left[s_a^\intercal \Theta_p^s c_u + x_a(t)^\intercal \Theta_p^x c_u - r(u,a,p) \right]^2 + \lambda \|x_a\|_2^2. \end{aligned}$$

To solve the ridge regression Eq. (4), we can easily derive the closed-form solutions as $x_{a,t} = \left(\Psi_{a,t}\right)^{-1} b_{a,t}$, where

$$\Psi_{a,t} = \sum_{u \in \mathcal{U}_{a,t}} \left(\Theta_p^x c_u\right) \left(\Theta_p^x c_u\right)^\intercal + \lambda I_{l_a},$$

$$b_{a,t} = \sum_{(u,a,r) \in \mathcal{H}_t} \left(\Theta_p^x c_u\right) \left[r(u,a,p) - \left(\Theta_p^s c_u\right)^\intercal s_a \right],$$

where $\mathcal{U}_{a,t}$ denotes the set of users (possibly with duplicates) who have been recommended item a until time t, and $I_{l_a} \in \mathbb{R}^{l_a \times l_a}$ is a identity matrix. The statistics $\left(\Psi_{a,t}, b_{a,t}\right)$ can be updated incrementally and the detailed computation can be found in Algorithm 1.

According to the UCB policy in bandit algorithms [22, 33, 41], we define the following UCB-based relevance score function for executing action (i.e., online recommendation) at time t:

$$f_t(c_u, c_a(t)) := \left[s_a^\intercal, x_a(t)^\intercal\right] \Theta_p c_u + \alpha \left[\left(\Theta_p^x c_u\right)^\intercal \left(\Psi_{a,t}\right)^{-1} \Theta_p^x c_u \right]^{\frac{1}{2}},$$

where $\alpha > 0$ is the exploration parameter, and the term multiplied by α is the exploration term. In this way, the executed action at time t can be selected by $a_{I_t} = \arg\max_{a \in \mathcal{A}_t} f_t(c_u, c_a(t))$.

4.2.2 Hypernetwork for Time-Varying Preference. In the last section, we describe the bandit policy given parameter matrix Θ_p in time period p. In this section, we explain how the hypernetwork generates the parameter matrix Θ_p. The main concept involves utilizing a hypernetwork that takes the embedding of the current time period as input and generates the parameters of the user preference matrix in the bandit policy. This enables the policy to adapt and adjust itself to accommodate changes in the distribution of user preferences over time.

Algorithm 1: HyperBandit

INPUT: Latent features of items $\boldsymbol{x}_{a\in\mathcal{A}} = \boldsymbol{0}^{l_a}$,data buffer $\mathcal{D}_{n=1} = \emptyset$, $\boldsymbol{\Phi}_{a\in\mathcal{A},t=1} = \boldsymbol{O}^{l_a\times l_a}$, $\boldsymbol{b}_{a\in\mathcal{A},t=1} = \boldsymbol{0}^{l_a}$, $\{T_n\}_{n\in[N]}$set of time steps in each updating part, regularization parameter $\lambda > 0$, exploration parameter $\alpha > 0$.

Initialize hypernetwork parameters $\boldsymbol{\xi}_{n=1}$ with Xavier Normal
for $n \in [N]$ **do**
  for $t = 1$ to T_n **do**
    Receive the user u_t and the time period embedding $\boldsymbol{s}_{p_t}$
    Obtain the set of candidate items $\mathcal{A}_t$
    Obtain the observed features $\boldsymbol{s}_a, \forall a \in \mathcal{A}_t$
    Obtain the latent features $\boldsymbol{x}_a(t), \forall a \in \mathcal{A}_t$
    Estimated the user preference matrix $\boldsymbol{\Theta}_{p_t}^{(n)} \coloneqq \left[\boldsymbol{\Theta}_{p_t}^{s(n)\intercal}, \boldsymbol{\Theta}_{p_t}^{x(n)\intercal}\right]^\intercal \leftarrow h_{\xi_n}(\boldsymbol{s}_{p_t})$
    Recommend item $a_{I_t} \in \mathcal{A}_t$ to user u_t following $a_{I_t} \leftarrow \arg\max_{a\in\mathcal{A}_t} \left[\boldsymbol{s}_a^\intercal, \boldsymbol{x}_a(t)^\intercal\right] \boldsymbol{\Theta}_{p_t}^{(n)} \boldsymbol{c}_{u_t} + \alpha \left[\left(\boldsymbol{\Theta}_{p_t}^{x(n)} \boldsymbol{c}_{u_t}\right)^\intercal \left(\boldsymbol{\Psi}_{a,t}\right)^{-1} \boldsymbol{\Theta}_{p_t}^{x(n)} \boldsymbol{c}_{u_t}\right]^{\frac{1}{2}}$
    Observe reward $r_t = r(u_t, a_{I_t}, p_t)$
    $\mathcal{D}_n \leftarrow \mathcal{D}_n \cup \left\{(u_t, a_{I_t}, p_t, r_t, \mathcal{A}_t)\right\}$
    // Bandit Policy Updating
    Get the user preference vector $\boldsymbol{P}_t \leftarrow \left(\boldsymbol{\Theta}_{p_t}^{x(n)} \boldsymbol{c}_{u_t}\right)^\intercal \in \mathbb{R}^{l_a}$ for the latent item features
    Get the user preference vector $\boldsymbol{Q}_t \leftarrow \left(\boldsymbol{\Theta}_{p_t}^{s(n)} \boldsymbol{c}_{u_t}\right)^\intercal \in \mathbb{R}^{o_a}$ for the observed item features
    $\boldsymbol{\Phi}_{a_{I_t},t+1} \leftarrow \boldsymbol{\Phi}_{a_{I_t},t} + \boldsymbol{P}_t^\intercal \boldsymbol{P}_t, \quad \boldsymbol{\Psi}_{a_{I_t},t+1} \leftarrow \lambda \boldsymbol{I} + \boldsymbol{\Phi}_{a_{I_t},t+1}$
    $\boldsymbol{b}_{a_{I_t},t+1} \leftarrow \boldsymbol{b}_{a_{I_t},t} + \boldsymbol{P}_t^\intercal \left(\boldsymbol{r}_t - \boldsymbol{Q}_t \boldsymbol{s}_{a_{I_t}}\right)$
    $\boldsymbol{x}_{a_{I_t},t+1} \leftarrow \left(\boldsymbol{\Psi}_{a_{I_t},t+1}\right)^{-1} \boldsymbol{b}_{a_{I_t},t+1}$
  end for
  // Hypernetwork Updating
  Update hypernetwork parameter $\boldsymbol{\xi}_{n+1} \leftarrow \Delta(\boldsymbol{\xi}_n)$ using efficient training method via low-rank factorization (in Sec. 4.3.2) and the Adam optimizer on $\mathcal{D}_n$
  Release $\mathcal{D}_n$ and set $\mathcal{D}_{n+1} \leftarrow \emptyset$
end for

To ensure stability in online recommendation, we incrementally update the hypernetwork h in mini-batches, where the total T time steps are divided into N parts, and the n-th part, $n \in [N]$, contains T_n time steps, corresponding to T_n interaction histories. In this way, the hypernetwork h is updated N times, and during the n-th update, the data buffer $\mathcal{D}_n \coloneqq \left\{(u_t, a_{I_t}, p_t, r_t, \mathcal{A}_t)\right\}_{t\in[T_n]}$ is used as the training data[1], where $r_t = r(u_t, a_{I_t}, p_t)$. Then, given the time period embedding $\boldsymbol{s}_p$, the hypernetwork after the $(n-1)$-th update, denoted by h_{ξ_n}, can be represented by:

$$\boldsymbol{\Theta}_p^{(n)} \coloneqq h_{\xi_n}(\boldsymbol{s}_p), \tag{5}$$

where ξ_n represents the model parameters of the hypernetwork, and the superscript (n) on $\boldsymbol{\Theta}_p^{(n)}$ indicates that it is generated by h_{ξ_n}. As illustrated in Figure 2, we implement the hypernetwork h using a Multi-Layer Perceptron (MLP). In this way, the MLP acts like a condition network, inputting the embedding of current time period, outputting the corresponding user preference matrix. Besides, the time period embedding $\boldsymbol{s}_p$ is generated via GloVe model [26] by imputing the current time period p.

To train the hypernetwork, we take inspiration from the Listnet loss design [3] to quantify the discrepancy between the estimated reward and the true label for each candidate item in $\mathcal{A}_t$ at time t. Specially, we use the estimated time-varying reward $\hat{r}_{u,a,p} = \boldsymbol{c}_a^\intercal \boldsymbol{\Theta}_p \boldsymbol{c}_u$ in Eq. (3), and construct the true labels according to the following rules: if the user u clicks on the recommended item a in time period p, the label is set to 1; if the user skips the recommended item, the label is set to -1; if the item is a candidate but not recommended, the label is set to 0. Formally, $y_{u,a,p} = 1$ if $r_{u,a,p} = 1$; $y_{u,a,p} = -1$ if $r_{u,a,p} = 0$; $y_{u,a,p} = 0$ if it is a candidate item but not recommended. Then, assuming that the number of actions is $M \coloneqq |\mathcal{A}_1| = \cdots = |\mathcal{A}_T|$, during the n-th incremental update, the loss function on the data buffer $\mathcal{D}_n$ could be shown as follows:

$$\mathcal{L}_\xi^{(n)} = -\sum_{t=1}^{T_n} \sum_{k=1}^{M} P\left(y_{u_t,a_k,p_t}\right) \log \widehat{P}\left(\hat{r}_{u_t,a_k,p_t}\right),$$

where ξ represents the hypernetwork parameters that need to be optimized, p_t corresponds to the time period where index t in $\mathcal{D}_n$ is located, and

$$\widehat{P}\left(\hat{r}_{u_t,a_k,p_t}\right) = \frac{\exp\left(\hat{r}_{u_t,a_k,p_t}\right)}{\sum_{i=1}^{M} \exp\left(\hat{r}_{u_t,a_i,p_t}\right)}, \; P\left(y_{u_t,a_k,p_t}\right) = \frac{\exp\left(y_{u_t,a_k,p_t}\right)}{\sum_{i=1}^{M} \exp\left(y_{u_t,a_i,p_t}\right)}.$$

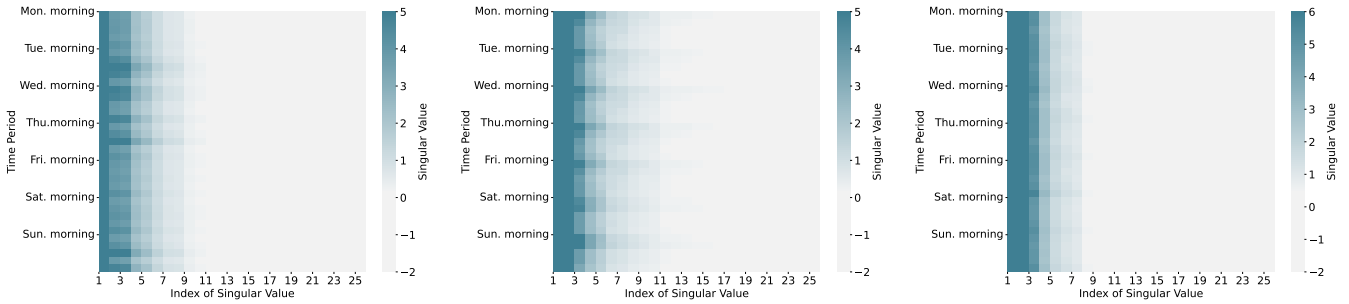
4.3 Efficient Training via Low-Rank Factorization

4.3.1 Analysis of Low-Rank Structure of User Preference Matrix. Since the user preference matrix $\boldsymbol{\Theta}_p \in \mathbb{R}^{d_a\times d_u}$ is generated by the hypernetwork in Eq. (5), a large output dimension (i.e., $d_a \times d_u$) would incur significant training costs. Hence, we consider representing the entire user preference matrix using a smaller number of parameters. Based on this motivation, it is natural to investigate whether the user preference matrix $\boldsymbol{\Theta}_p$ exhibits a low-rank structure. To verify the presence of low-rank structures, we perform singular value decomposition (SVD) on $\boldsymbol{\Theta}_p^{(N)}$ across different time periods. As shown in Fig. 3, it is apparent that the distribution of singular values is concentrated in the first few dimensions, which is far less than the dimensionality of the user preference matrices[2]. Hence, we deduce the existence of low-rank structures within the user preference matrices, suggesting that a low-rank representation of the matrix $\boldsymbol{\Theta}_p$ effectively preserves the majority of its informational content.

4.3.2 Training Process with Low-Rank Factorization. Based on the analysis above, we try to improve the training efficiency of hypernetwork through explicitly modeling the low-rank structure in the user preference matrix $\boldsymbol{\Theta}_p$ (for ease of exposition, we omit the superscript of $\boldsymbol{\Theta}_p^{(n)}$ below). Specifically, we propose to approximate $\boldsymbol{\Theta}_p$ with its low-rank approximation. Here, we leverage matrix factorization approach to achieve the approximation. Given the *estimated rank* $\tau > 0$, we model the low-rank structure of $\boldsymbol{\Theta}_p$ with the

[1] It is important to note that the interaction history corresponding to the same index t in different data buffers $\mathcal{D}_n$ may be different.

[2] Following the setting of baselines [16, 22, 33, 36], we set $d_a = d_u = 25$, thus $\boldsymbol{\Theta}_p$ is a square matrix.



(a) KuaiRec (b) NYC (c) TKY

Figure 3: The distribution of singular eigenvalues (SEs) of user preference matrices across different time periods. The horizontal axis represents the index of SEs, arranged in descending order, while the vertical axis represents the time periods. The darkness of the colors corresponds to the magnitude of the singular values.

product of two rank-τ latent matrices $\boldsymbol{A}_p \in \mathbb{R}^{d_a \times \tau}$ and $\boldsymbol{B}_p \in \mathbb{R}^{d_u \times \tau}$, i.e., $\boldsymbol{\Theta}_p \approx \boldsymbol{A}_p \boldsymbol{B}_p^\intercal$, as shown in Fig. 4.

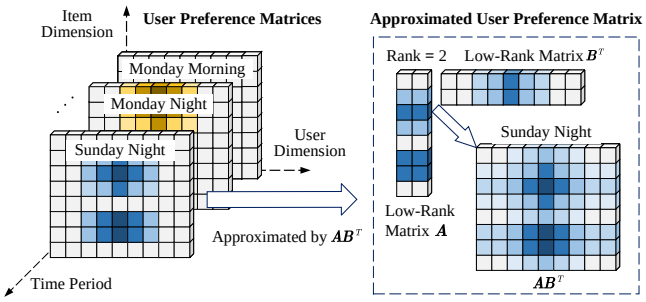


Figure 4: User preference matrix estimation using low rank factorization: An example with estimated rank $\tau = 2$.

In the implementation of the hypernetwork, given a time period $p \in \mathcal{P}$, the hypernetwork outputs a vector represented by $\text{Concat}(\text{Vec}(\boldsymbol{A}_p), \text{Vec}(\boldsymbol{B}_p)) \in \mathbb{R}^{\tau d_a + \tau d_u}$, where $\text{Concat}(\cdot)$ denotes the concatenation operation. Then, the vector $\text{Vec}(\boldsymbol{A}_p) \in \mathbb{R}^{\tau d_a}$ is reshaped into a matrix $\boldsymbol{A}_p \in \mathbb{R}^{d_a \times \tau}$, and the vector $\text{Vec}(\boldsymbol{B}_p) \in \mathbb{R}^{\tau d_u}$ is reshaped into a matrix $\boldsymbol{B}_p \in \mathbb{R}^{d_u \times \tau}$. Finally, the product $\boldsymbol{A}_p \boldsymbol{B}_p^\intercal$ is obtained to estimate $\boldsymbol{\Theta}_p$. This matrix factorization reduces the output dimension of the hypernetwork h (defined in Eq. (5)) from $d_a d_u$ to $\tau(d_a + d_u)$, effectively alleviating the training efficiency issues.

5 REGRET ANALYSIS

The regret bound serves as a fundamental theoretical guarantee for online learning algorithms [2, 5, 18, 27, 42]. In this section, we provide a regret bound of the proposed HyperBandit. First, we define the *regret* as follows:

$$\text{Reg}(T) := \sum_{t \in [T]} \left[r^*(u_t, a_t^*, p_t) - r^*(u_t, a_{I_t}, p_t) \right], \tag{6}$$

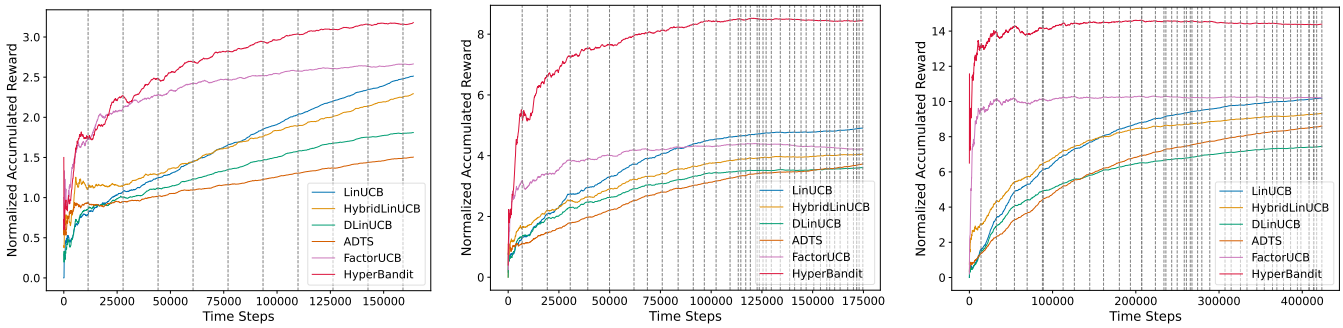
where a_t^* represents the action with the highest time-varying true reward r^* (defined in Eq. (2)) at time t, u_t denotes the user for whom the item is recommended at time t, and p_t represents the time period to which t belongs. Recalling that I_t denotes the index of the action executed by HyperBandit at time t, the regret in Eq. (6) measures the difference between the accumulated time-varying true rewards of the best policy and our policy.

Theorem 5.1 (Regret Upper Bound of HyperBandit). *Assume that the dimension of the latent features is $l_a = L, \forall a \in \mathcal{A}$. The sequence of the actions executed by HyperBandit enjoys the following regret upper bound: with probability at least $1 - \delta$,*

$$\text{Reg}(T) \leq 2C_{\boldsymbol{x}} \sqrt{2LT \ln\left(1 + \frac{C_{\boldsymbol{\Theta}} C_{\mathcal{U}}^2 T}{2\lambda L \delta}\right)} + C_{\boldsymbol{x}} \sqrt{\frac{2}{\lambda}} \sum_{n \in [N]} E_n, \tag{7}$$

where 1). $C_{\boldsymbol{x}} = \max_{a \in \mathcal{A}_T} \|\boldsymbol{x}_a(T) - \boldsymbol{x}_a^*\|_{\boldsymbol{\Psi}_{a,T}}$, $\|\boldsymbol{x}\|_{\boldsymbol{\Psi}} := \sqrt{\boldsymbol{x}^\intercal \boldsymbol{\Psi} \boldsymbol{x}}$ denotes the elliptic norm of $\boldsymbol{x}$ with respect to the matrix $\boldsymbol{\Psi}$, $\boldsymbol{x}_a^*$ is the true latent features of item a, and $C_{\boldsymbol{\Theta}} = \max_{n \in [N], p \in \mathcal{P}} \left\| \boldsymbol{\Theta}_p^{(n)} \right\|_{\text{F}}^2$, $C_{\mathcal{U}} = \max_{u \in \mathcal{U}} \|\boldsymbol{c}_u\|_2^2$; 2). $E_n := \sum_{i \in [T_n]} \left\| \left(\boldsymbol{\Theta}_{p_{i,n}}^{(n)} - \boldsymbol{\Theta}_{p_{i,n}}^* \right) \boldsymbol{c}_{u_{i,n}} \right\|_2$ denotes the error caused by the hypernetwork updated N times using T_n examples in the n-th update, and the subscript $(\cdot)_{i,n}$ denotes the time step $i \in [T_n]$ after the $(n-1)$-th update.

The error E_n in Eq. (7) caused by the hypernetwork can be decomposed into the sum of the following three parts. 1). **Approximation error** measures the discrepancy between the optimal hypothesis in hypernetwork space and the target function that generates $\boldsymbol{\Theta}_p^* \boldsymbol{c}_u$ in E_n. From the results in [10], we obtain that the approximation error of the hypernetwork h (defined in Eq. (5)) with ReLU activation function is ε, providing that the number of trainable parameters in the hypernetwork is $\Omega\left(\varepsilon^{-U/S^*} + \varepsilon^{-P/S^*}\right)$, where we assume $d_u = U, \forall u \in \mathcal{U}$, $d_p = P, \forall p \in \mathcal{P}$, and S^* denotes the order of smoothness of the target function. 2). **Optimization error**: measures the accumulated deviation between the hypernetwork parameters obtained through the online optimization algorithm and those of the optimal hypothesis in the hypernetwork space. Our HyperBandit equipped with a mini-batch first-order optimization method incurs an accumulated optimization error of order $O(T/N)$, assuming that each data buffer $\mathcal{D}_n, n \in [N]$ contains an equal number of examples. 3). **Estimation error** measures the the



(a) KuaiRec (b) NYC (c) TKY

Figure 5: Normalized accumulated reward of baselines, and the proposed HyperBandit on three datasets, KuaiRec & NYC & TKY. Note that The grey dashed lines represent the boundaries between weekdays and weekends. The x-axis represents the interaction data arranged in chronological order, and the y-axis represents the normalized accumulated reward.

error caused by the estimated user preference matrix using low-rank factorization. According to the analyses in Sec. 4.3.1, the user preference matrix exhibits a low-rank structure. Assuming that the maximum rank of the user preference matrices is R, if the estimated rank τ in the low-rank factorization discussed in Sec. 4.3.2 is set to $\tau \geq R$, and the best rank-τ approximation can be obtained, then the estimation error would be zero.

Setting the number of trainable parameters in the hypernetwork as $\Omega\left(\sqrt{T}^{U/S^*} + \sqrt{T}^{P/S^*}\right)$, the number of hypernetwork training iterations $N = O(\sqrt{T})$, and the estimated rank $\tau \geq R$, we can derive an upper bound for the error term $\sum_{n\in[N]} E_n$ in Eq. (7) of order $\widetilde{O}(\sqrt{T})$. This, in turn, leads to a sublinear regret upper bound of order $\widetilde{O}(\sqrt{T})$ for HyperBandit.

6 EXPERIMENTS

We conducted experiments to evaluate the performance of HyperBandit on short video recommendation and point-of-interest (POI) recommendation.

6.1 Experimental Settings

6.1.1 Baselines. HyperBandit was compared with several algorithms that constructed in stationary or piecewise-stationary environment:

LinUCB [22] is a classical contextual bandit algorithm that addresses the problem of personalized recommendation.

HybridLinUCB [22] is a variant algorithm of LinUCB that takes into account both shared and non-shared interests among users.

DLinUCB [36] is built upon a piecewise stationary environment, where each user group corresponds to a slave model. Whether to discard a slave model is based on the detection of "badness".

ADTS [16] is a bandit algorithm based on Thompson sampling, which tend to discard parameters before the last change point.

FactorUCB [33] leverages observed contextual features and user interdependencies to improve the convergence rate and help conquer cold-start in recommendation.

6.1.2 Hyperparameter Settings. We implemented the hypernetwork h in Eq. (5) using a MLP. The MLP consists of 1 input layer, 8 hidden layers, and 1 output layer. The number of nodes in the each layer is as follows: 30, 256, 512, 1024, 1024, 1024, 1024, 512, 256, $25 * \tau * 2$. We applied ReLU activation function after each hidden layer. We trained the hypernetwork every 2000 time steps (i.e., $T_n = 2000, n \in [N]$) on KuaiRec and NYC, while $T_n = 5000$ on TKY. Early stopping is applied in training process to avoid overfitting.

For the parameters in bandit policy, we set the exploration parameter α to 0.1 and the regularization parameter λ to 0.1 for all the algorithms. The size of the candidate item set $\mathcal{A}_t, \forall t \in [T]$ at each time step was set to 25 in all algorithms. The dimensions of both the context features of users and the context features of items were set to 25. In FactorUCB and HyperBandit, the dimensions of latent features of items were set to 10, and the dimensions of observed features of items were set to 15.

6.1.3 Evaluation Protocol. The accumulated reward (AR) was utilized to assess the recommendation accuracy of algorithms, which denotes the sum of the observed reward from the beginning to the current step. The normalized accumulated reward refers to the AR normalized by the corresponding logged random strategy.

6.2 Experiments on Short Video Recommendation

We employed KuaiRec[3] for evaluation, that is a real-world dataset [11] collected from the recommendation logs of the video-sharing mobile app Kwai. The dense interaction matrix we used contains 1411 users, 3327 items and 529 video categories (i.e., tags). Following the settings in [32], we used video categories (tags) as actions. In this experiment, we treated watch ratio higher than 2.0 as positive feedback. If the action received positive feedback in other time periods but not in the current period, we assumed that the current user would give negative feedback to it.

To fit the data into the contextual bandit setting, we pre-processed it first. We encoded all the user features provided by KuaiRec, which contain information like user activity level, number of followed users and others. Subsequently, we applied PCA to reduce the dimensionality of the context feature vectors. We retained the first 25 principal components and applied the same procedure to attain the context feature vectors of items (i.e., $d_a = d_u = 25$). For a particular time step, the video tag having positive feedback was picked and the

[3]https://github.com/chongminggao/KuaiRec

Table 2: Comparisons of normalized accumulated reward, running time (sec., mean) and training time (sec., mean) of hypernetwork on KuaiRec, Foursquare (NYC) and Foursquare (TKY). The "Running Time of BP" means the average time cost of online recommendation and updating by Bandit Policy at each time step, and the "Training Time of HN" means the average time cost for training HyperNetwork at each time step. "–" means the corresponding algorithm has no hypernetwork.

Algorithm	Normalized Accumulated Reward			Running Time of BP		Training Time of HN	
	KuaiRec	NYC	TKY	KuaiRec	Foursquare	KuaiRec	Foursquare
LinUCB	2.56 ± 0.04	4.86 ± 0.05	10.15 ± 0.10	3.09e−04	3.08e−04	–	–
HybridLinUCB	2.29 ± 0.03	4.05 ± 0.07	9.33 ± 0.03	2.52e−02	2.65e−02	–	–
DLinUCB	1.84 ± 0.03	3.63 ± 0.08	7.39 ± 0.08	3.51e−04	3.55e−04	–	–
ADTS	1.50 ± 0.02	3.80 ± 0.10	8.63 ± 0.09	6.52e−03	6.69e−03	–	–
FactorUCB	2.70 ± 0.05	4.19 ± 0.04	10.22 ± 0.03	1.01e−01	1.11e−01	–	–
HyperBandit ($\tau = 1$)	**3.79 ± 0.18**	6.46 ± 0.45	13.37 ± 0.22	1.65e−03	1.62e−03	1.45e−04	1.06e−04
HyperBandit ($\tau = 5$)	3.51 ± 0.06	8.08 ± 0.09	13.99 ± 0.29	1.60e−03	1.63e−03	1.58e−04	1.21e−04
HyperBandit w/o Low-Rank	3.24 ± 0.11	**8.27 ± 0.17**	**14.49 ± 0.07**	1.63e−03	1.62e−03	1.72e−04	1.73e−04

remaining 24 were randomly sampled from the tags which would get negative feedback in the current time step. Besides, we extracted data for one week from August 10th, 2020, to August 16th, 2020. From each time period within that week, we randomly sampled several events to reconstruct the data. This sampling process was repeated 10 times to generate 10 weeks' data.

As shown in Fig. 5a and Table 2, HyperBandit outperformed all the other baselines on KuaiRec in terms of rewards. Both DLinUCB and ADTS were worse than others since these two algorithms were designed for the piecewise stationary environment (i.e., they abandon the knowledge acquired during past periods instead of utilizing). As more observations recurrent, LinUCB quickly catched up, because it is better to regard periodic environment as a stationary environment rather than piecewise environment. Furthermore, FactorUCB utilizes observed contextual features and interdependencies among users to enhance the algorithm's convergence rate, resulting in strong initial performance. However, this improvement comes at the cost of increased time consumption, as it necessitates updating all user parameters in each time step.

6.3 Experiments on POI Recommendation

Foursquare NYC & TKY dataset [38] includes long-term (about 10 months) check-in data in New York city (NYC) and Tokyo (TKY) collected from Foursquare[4] from 12 April 2012 to 16 February 2013. Table 3 shows the statistics of two check-in datasets: NYC and TKY.

Table 3: The statistics of the Foursquare NYC & TKY.

Dataset	#Users	#POIs	#POI Categories	#Check-ins
NYC	1,083	38,333	400	227,428
TKY	2,293	61,858	385	573,703

Similarly, we used POI categories as actions. The ground-truth categories of the check-ins were considered positive samples of the current step while the rest categories were considered negative. Initially, instead of following the setting in [4] that used TF-IDF for representation construction, we employed the GloVe model [26] to craft a 300-dimensional feature vector, enhancing item context representation. Subsequently, PCA reduced vector dimensionality, retaining the initial 25 principal components. Since these two datasets do not contain user profiles, we used the interaction data from the first week to construct the contextual features of users. Specifically, we used the average vector of all the POI category feature vectors that the user checked in during the first week as the user context feature vector. For the candidate action set at each time step, we selected the ground-truth check-in tag and randomly extracted 24 negative categories of the current step. We used all the data from April 10th, 2012 to February 16th, 2013 (except data from the first week) to construct a data streaming.

As illustrated in Fig. 5b and Fig. 5c, similar conclusions can be drawn as in KuaiRec. Additionally, we conducted an analysis of time cost for all algorithms and compared the performance of different estimated rank ($\tau = 1$, $\tau = 5$ and w/o Low-Rank) of HyperBandit. The corresponding results are presented in Table 2. Notably, HyperBandit consistently outperformed the baselines in terms of normalized accumulated reward, while the running time of BP (bandit policy) remained acceptable. Furthermore, HyperBandit ($\tau = 1$) and HyperBandit ($\tau = 5$) reduced the training time of HN by 15.7%, 8.1% in KuaiRec and 38.7%, 30.1% on Foursquare dataset compared to HyperBandit (w/o Low-Rank), which provided strong evidences of the training efficiency with low-rank factorization.

6.4 Ablation Experiments

In this section, we empirically studied the proposed HyperBandit by addressing the following research questions:

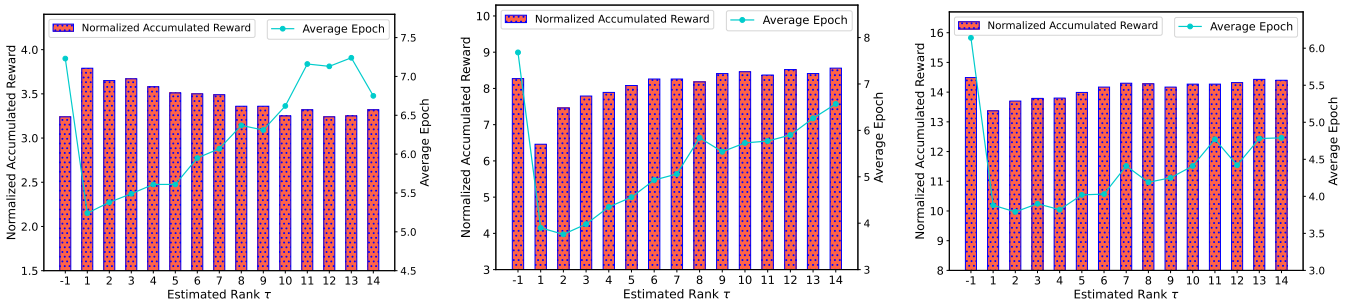
RQ1: Is HyperBandit efficient enough to meet the real-time requirements of online recommendations?

RQ2: How does the estimated rank τ of low-rank factorization affect HyperBandit?

RQ3: What is the impact of key components in HyperBandit on the recommendation performance?

6.4.1 ***RQ1: Running Time.*** In streaming recommendation scenarios, running time is another important metric. We reports the running time of bandit policy and the training time of hypernetwork in Table 2. From the results, we conclude that the time cost of HyperBandit was on the order of milliseconds (ms), indicating that

[4]https://foursquare.com/



(a) KuaiRec (b) NYC (c) TKY

Figure 6: Performance of low-rank factorization in HyperBandit on different estimated rank across three datasets. Note that the bar chart shows normalized accumulated reward, while the line chart shows average epoch in training process (a larger average epoch indicates a longer training time of hypernetwork). The first data point (with an x-coordinate of "-1") represents the result obtained without utilizing low-rank factorization.

HyperBandit met the real-time requirements in streaming recommendations. Furthermore, the training time of the hypernetwork exhibits a decreasing trend as the estimated rank τ decreases from full rank to 1, validating its efficiency in low-rank updating.

6.4.2 ***RQ2: Impact of Estimated Rank*** τ. Fig. 6 explored the impact of different estimated rank of low-rank factorization on the performance in terms of normalized accumulated reward and training time. The observations from the experimental results can be summarized as follows: 1). With an increase in the estimated rank τ, our HyperBandit demonstrated an overall improvement in normalized accumulated reward on Foursquare datasets. Furthermore, HyperBandit with ranks from 1 to 9 even outperformed the algorithm without low-rank factorization in terms of rewards on the KuaiRec dataset. These results validate the effectiveness of the low-rank factorization approach, which maintains excellent performance. 2). The average epoch, which measures the training time of the hypernetwork, also exhibits an overall upward trend as the estimated rank τ increases, although it is significantly smaller than that without low-rank factorization. This observation highlights the benefits of efficient training via low-rank factorization as described in Sec. 4.3.2.

6.4.3 ***RQ3: Impact of Key Components in HyperBandit.*** HyperBandit consists of two key components for online updating: one is to update the latent features of items via ridge regression, and the other is to update the parameters of the hypernetwork through gradient descent. To investigate the interplay between these two updating components, an ablation experiment was conducted with the following settings: 1). Disable ridge regression updating: The dimension of the latent item features was set to zero. 2). Disable hypernetwork updating: The hypernetwork parameters were frozen to their initial state. The results are presented in Table 4. Based on the results, the following conclusions can be drawn: 1). Employing both updating components independently enhances the recommendation performance. 2). Irrespective of whether ridge regression was enabled or disabled, the utilization of the hypernetwork can lead to performance improvements.

Table 4: The results of the ablation experiment on key components of HyperBandit. The ridge regression updating in the bandit policy is denoted by "RR", and the hypernetwork updating is denoted by "HN". The symbol ✓ signifies the inclusion of a particular update process, while the symbol ✗ indicates its exclusion.

RR	HN	Normalized Accumulated Reward		
		KuaiRec	NYC	TKY
✗	✗	0.93 ± 0.08	0.87 ± 0.08	1.14 ± 0.44
✓	✗	1.09 ± 0.08	5.62 ± 0.52	13.26 ± 0.23
✗	✓	1.86 ± 0.09	4.90 ± 0.28	10.91 ± 0.35
✓	✓	**3.24 ± 0.11**	**8.27 ± 0.17**	**14.49 ± 0.07**

7 CONCLUSION

This paper aims to model the user preference shift in periodic non-stationary streaming recommendation scenarios. Specifically, we propose an online learning approach called HyperBandit. The proposed HyperBandit leverages a hypernetwork to dynamically adjust user preference parameters for estimating time-varying rewards, employs a bandit policy for online recommendation with a regret guarantee, and utilizes a low-rank factorization method to efficiently train the model. Experimental results demonstrated the effectiveness and efficiency of HyperBandit in steaming recommendation. The proposed HyperBandit has opened up a promising avenue for advancing controllable online learning.

ACKNOWLEDGMENTS

This work was funded by the National Key R&D Program of China (2022ZD0114802), Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. Supported by fund for building world-class universities (disciplines) of Renmin University of China. Supported by Public Computing Cloud, Renmin University of China. Supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (23XNKJ13).

REFERENCES

[1] Santiago Balseiro, Negin Golrezaei, Mohammad Mahdian, Vahab Mirrokni, and Jon Schneider. 2019. Contextual Bandits with Cross-Learning. In *Advances in Neural Information Processing Systems 32*. 9679–9688.

[2] Sébastien Bubeck and Nicolò Cesa-Bianchi. 2012. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Journal of Foundations and Trends in Machine Learning* 5 (2012), 1–122.

[3] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine learning*. 129–136.

[4] Nicolo Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A gang of bandits. (2013), 2265–2279.

[5] Nicolò Cesa-Bianchi and Gabor Lugosi. 2006. *Prediction, learning, and games*. Cambridge University Press.

[6] Badrish Chandramouli, Justin J Levandoski, Ahmed Eldawy, and Mohamed F Mokbel. 2011. Streamrec: A real-time recommender system. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. 1243–1246.

[7] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang. 2017. Streaming Recommender Systems. In *Proceedings of the 26th International Conference on World Wide Web*. 381–389.

[8] Sirui Chen, Yuan Wang, Zijing Wen, Zhiyu Li, Changshuo Zhang, Xiao Zhang, Quan Lin, Cheng Zhu, and Jun Xu. 2023. Controllable Multi-Objective Re-ranking with Policy Hypernetworks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3855–3864.

[9] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. 2015. Learning in nonstationary environments: A survey. *Journal of IEEE Computational Intelligence Magazine* 10 (2015), 12–25.

[10] Tomer Galanti and Lior Wolf. 2020. On the modularity of hypernetworks. (2020), 10409–10419.

[11] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A Fully-Observed Dataset and Insights for Evaluating Recommender Systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 540–550.

[12] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM conference on Recommender Systems*. 93–100.

[13] Aurélien Garivier and Eric Moulines. 2008. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415* (2008).

[14] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[15] Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose Blanchet, Peter W Glynn, and Yinyu Ye. 2020. Sequential batch learning in finite-action linear contextual bandits. *arXiv preprint arXiv:2004.06321* (2020).

[16] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting to user preference changes in interactive recommendation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. 4268–4274.

[17] C. Hartland, S. Gelly, N. Baskiotis, O. Teytaud, and M. Sebag. 2006. Multi-armed Bandit, Dynamic Environments and Meta-Bandits. (2006).

[18] Elad Hazan. 2016. Introduction to online convex optimization. *Journal of Foundations and Trends in Optimization* 2 (2016), 157–325.

[19] Martin Jakomin, Zoran Bosnic, and Tomaz Curk. 2020. Simultaneous Incremental Matrix Factorization for Streaming Recommender Systems. *Journal of Expert Systems with Applications* 160 (2020), 113685.

[20] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. 2019. Hypernetwork functional image representation. In *Proceedings of 28th International Conference on Artificial Neural Networks*. 496–510.

[21] John Langford and Tong Zhang. 2007. The epoch-greedy algorithm for contextual multi-armed bandits. (2007), 96–1.

[22] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 661–670.

[23] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. 2019. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv preprint arXiv:1903.03088* (2019).

[24] Eliya Nachmani and Lior Wolf. 2019. Hyper-graph-network decoders for block codes. (2019), 2326–2336.

[25] Aviv Navon, Aviv Shamsian, Ethan Fetaya, and Gal Chechik. 2021. Learning the Pareto Front with Hypernetworks. In *Proceedings of the 9th International Conference on Learning Representations*.

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[27] Shai Shalev-Shwartz. 2011. Online learning and online convex optimization. *Journal of Foundations and Trends in Machine Learning* 4 (2011), 107–194.

[28] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *Proceedings of the 38th International Conference on Machine Learning*. 9489–9502.

[29] Aleksandrs Slivkins and Eli Upfal. 2008. Adapting to a Changing Environment: the Brownian Restless Bandits.. In *COLT*. 343–354.

[30] Joseph Suarez. 2017. Language modeling with recurrent highway hypernetworks. *Advances in neural information processing systems* 30 (2017), 3267–3276.

[31] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. 2020. Continual learning with hypernetworks. In *Proceedings of the 8th International Conference on Learning Representations*.

[32] Yongquan Wan, Junli Xian, and Cairong Yan. 2021. A Contextual Multi-armed Bandit Approach Based on Implicit Feedback for Online Recommendation. In *Proceedings of the 15th Knowledge Management in Organizations*. 380–392.

[33] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2017. Factorization bandits for interactive recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[34] 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.

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

[36] Qingyun Wu, Naveen Iyer, and Hongning Wang. 2018. Learning contextual bandits in a non-stationary environment. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 495–504.

[37] Xiao Xu, Fang Dong, Yanghua Li, Shaojian He, and Xin Li. 2020. Contextual-bandit based personalized recommendation with time-varying user interests. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 6518–6525.

[38] Dingqi Yang, Daqing Zhang, Vincent. W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *Journal of IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45 (2015), 129–142.

[39] Jia Yuan Yu and Shie Mannor. 2009. Piecewise-stationary bandit problems with side observations. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1177–1184.

[40] Xiao Zhang, Sunhao Dai, Jun Xu, Zhenhua Dong, Quanyu Dai, and Ji-Rong Wen. 2022. Counteracting user attention bias in music streaming recommendation via reward modification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2504–2514.

[41] Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. 2021. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 41–50.

[42] Xiao Zhang, Yun Liao, and Shizhong Liao. 2019. A survey on online kernel selection for online kernel learning. *WIREs Data Mining and Knowledge Discovery* 9 (2019), e1295.