Generalized Contextual Bandits With Latent Features: Algorithms and Applications

Xiongxiao Xu, Hong Xie, *Member, IEEE*, and John C. S. Lui, *Fellow, IEEE*

***Abstract*— Contextual bandit is a popular sequential decision-making framework to balance the *exploration* and *exploitation* tradeoff in many applications such as recommender systems, search engines, etc. Motivated by two important factors in real-world applications: 1) latent contexts (or features) often exist and 2) feedbacks often have humans in the loop leading to human biases, we formulate a generalized contextual bandit framework with latent contexts. Our proposed framework includes a two-layer probabilistic interpretable model for the feedbacks from human with latent features. We design a *GCL-PS* algorithm for the proposed framework, which utilizes posterior sampling to balance the *exploration* and *exploitation* tradeoff. We prove a sublinear regret upper bound for *GCL-PS*, and prove a lower bound for the proposed bandit framework revealing insights on the optimality of *GCL-PS*. To further improve the computational efficiency of *GCL-PS*, we propose a Markov Chain Monte Carlo (MCMC) algorithm to generate approximate samples, resulting in our *GCL-PSMC* algorithm. We not only prove a sublinear Bayesian regret upper bound for our *GCL-PSMC* algorithm, but also reveal insights into the tradeoff between computational efficiency and sequential decision accuracy. Finally, we apply the proposed framework to hotel recommendations and news article recommendations, and show its superior performance over a variety of baselines via experiments on two public datasets.**

***Index Terms*— Generalized contextual bandit, latent features, Markov Chain Monte Carlo (MCMC), posterior sampling.**

I. Introduction

SEQUENTIAL decision making is quite common in many web or mobile applications. For example, video sites like Youtube recommend several videos to users for each of their video search. Online social networks (OSNs) like Facebook send multiple advertisements to users on a per session basis. News apps like Flipboard push news to users every hour. Recommender systems like IMDB recommend movies to users per login. Each time when a keyword is entered, search engines like Google select items to place on the first page. One common nature of these sequential decision-making problems is that the decision maker (e.g., IMDB, Youtube, etc.) needs to infer users' preference or tastes, etc., which are unknown to the decision maker. To illustrate, consider the following example.

Example 1: Consider a search engine in an E-commerce system like Amazon, eBay, etc., and the search keyword is "action movie." For simplicity, consider two items denoted by A and B. The keyword "action movie" is entered 100 times. Users who enter the keyword are uniformly sampled from the user population interested in action movies. Each time the search engine needs to select one item from A or B to place on the first page (for simplicity, assume the first page can place only one item). Suppose that the true click rates of placing A and B on the first page are 0.1 and 0.05, respectively, which are unknown to the search engine. The objective is to maximize the total number of clicks in these 100 searches of *"action movie"*.

One challenge of the problem stated above is how to make a balance between *exploitation* (i.e., place the item which currently has the highest empirical average click rate on the first page) and *exploration* (i.e., give the opportunity to the item which was less frequently placed on the first page) in each search of the keyword "action movie." One typical approach to address this challenge is via the upper confidence bound (UCB) algorithm [1]. The idea of the UCB algorithm is to place the item A (B) in the first (second) search of *action movie* in the first page, then for the third to 100th search of *action movie*, it selects the item with the largest index defined as follows:

$$\text{index}(x) = \text{EmRate}(x) + \text{Penalty}(x) \quad \forall x \in \{A, B\}.$$

Here, $\text{EmRate}(x)$ denotes the current empirical average click rate of item $x \in \{A, B\}$. The $\text{Penalty}(x)$ denotes a penalty for item x and it is decreasing in the number of times that an item was placed on the first page. The purpose of $\text{Penalty}(x)$ is to give some opportunities (i.e., placing on the first page) to the item which was rarely placed on the first page. However, the UCB algorithm is not efficient when there are many items to consider because it needs to give a sufficient number of opportunities to each item in consideration (i.e., place it on the first page). To illustrate, consider the following example.

Manuscript received December 17, 2020; revised June 19, 2021; accepted October 22, 2021. The work of H. Xie was supported in part by the National Natural Science Foundation of China under Grant 61902042 and in part by the Chongqing Natural Science Foundation under Grant cstc2020jcyj-msxmX0652. The work of J. C. S. Lui was supported in part by the General Research Fund (GRF) under Grant 14200420. *(Corresponding author: Hong Xie.)*

Xiongxiao Xu and Hong Xie are with the Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing 400044, China (e-mail: xuxiongxiao@gmail.com; xiehong2018@foxmail.com).

John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3124603.

Digital Object Identifier 10.1109/TNNLS.2021.3124603

Example 2: Consider the same setting as in Example 1, but we vary the number of items from 2 to 100. Furthermore, the true click rate of each item $i \in \{1, \ldots, 100\}$ has a linear form of $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\theta}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ denotes the feature vector of item i (each item has d features) and $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the collective preference vector for the user population interested in action movies. The $\boldsymbol{x}_i$ is known to the search engine, while the $\boldsymbol{\theta}$ is unknown. The objective is to select items to maximize the total number of clicks in 100 searches of *action movie*.

Using the above UCB algorithm, it will place each of these 100 items one by one on the first page in these 100 searches, implying poor decision making. One typical approach to fix this issue is via the LinUCB algorithm [2]. The algorithm takes advantage of the linear form $\boldsymbol{x}_i^T\boldsymbol{\theta}$ of the true click rate and learns about the preference vector $\boldsymbol{\theta}$. Once $\boldsymbol{\theta}$ is learned, the click rate of each item can be estimated as well. However, in practice, there usually exists some unobserved or latent features, under which the LinUCB algorithm may fail. To illustrate, consider the following example.

Example 3: Consider the same setting as Example 2, but each item i is associated with a latent feature $\boldsymbol{y}_i \in \mathbb{R}^\ell$, and $\boldsymbol{y}_i$ is associated with a preference vector $\boldsymbol{\vartheta} \in \mathbb{R}^\ell$. The true click rate is $\boldsymbol{x}_i^T\boldsymbol{\theta} + \boldsymbol{y}_i^T\boldsymbol{\vartheta}$.

In Example 3, the LinUCB has a risk of making wrong decisions because it only learns $\boldsymbol{\theta}$, while latent features are quite common in real-world Web applications [3]. *How to make sequential decisions (e.g., place items) when there are latent features?* Wang *et al.* [3] also considered a contextual bandit with latent feature, but their model considers only the linear reward function. Zhou and Brunskill [4] proposed a latent contextual bandit model, which associate each user with a latent type. Lamprier *et al.* [5] proposed a latent contextual bandit model that in each round the contextual vector of each action is generated by a normal distribution. All these work assume a linear reward function (or linear true click rate in the above examples). However, in real-world Web applications, the true click rate may not take the simple linear form $\boldsymbol{x}_i^T\boldsymbol{\theta} + \boldsymbol{y}_i^T\boldsymbol{\vartheta}$, as there are humans in the loop leading to human bias in the clicking behavior. *How to formulate a more realistic feedback model for Web applications?* A number of works considered nonlinear feedback models without latent features [6]–[10], and they demonstrated that going beyond the linear form makes it more challenging to make the right decision as well even without latent features. One needs to make a balance between the model complexity and the mathematical or computational tractability of the model. To address these challenges, we propose a generalized contextual bandit framework with latent features. More importantly, we develop efficient posterior sampling algorithms to balance the *exploration* and *exploitation* tradeoff. One challenge in the development of our framework is: *It is computationally expensive to generate samples from the posterior distribution. Our contributions are the following.*

1) We propose a two-layer probabilistic and interpretable model for feedbacks from humans. In the first layer, we model how an agent forms natural evaluation, which offers a flexible way to capture human factors like bias and latent features. In the second layer, we model how an agent maps its natural evaluation to feedback set specified by a Web application. Then we formulate a generalized contextual bandit framework with latent features and generalized linear reward functions.
2) We design a *GCL-PS* algorithm for the proposed framework, which utilizes posterior sampling to balance the *exploration* and *exploitation* tradeoff. We prove a sublinear regret upper bound for *GCL-PS*, and prove a lower bound for the proposed bandit framework revealing insights into the optimality of *GCL-PS*. We show that the *GCL-PS* algorithm has a high computational cost in generating samples from the posterior distribution.
3) To improve the computational efficiency, we propose an MCMC algorithm to generate approximate samples, leading to our *GCL-PSMC* algorithm. We not only prove that the *GCL-PSMC* achieves a *sublinear Bayesian regret upper bound*, but also reveal insights on the tradeoff between computational efficiency and sequential decision accuracy. More importantly, this tradeoff guides us to determine the stopping condition for the MCMC algorithm.
4) We conduct extensive experiments on synthetic data to evaluate the performance of the *GCL-PSMC* algorithm. Experimental results validate the superior performance of the *GCL-PSMC* algorithm over a variety of other algorithms. We also apply our framework to hotel recommendation and news article recommendation. Experiments on two public datasets further confirm the superior performance of the *GCL-PSMC* over a variety of baselines.

The remainder of this article is organized as follows. Section II presents the related work. Section III presents the model. Section IV presents the design and analysis of algorithms. Sections V and VI presents the experiments on synthetic data and real-world data, respectively. Section VII concludes.

II. Related Work

To the best of our knowledge, this is the first article to study generalized contextual bandit with latent features and generalized linear reward functions.

A. Contextual Bandit Model and Its Variants

Contextual bandit [2], [11] is a popular sequential decision making framework to balance the exploration and exploitation tradeoff in many applications such as recommender systems, search engines, etc. A variety of contextual bandit variants have been proposed. In the classical contextual bandit [2], [11] the expected reward (i.e., reward function) of each action is a linear function of the contexts and the preference vector. Filippi *et al.* [6] extended the classical contextual bandit model to consider a generalized linear reward function, which is a nonlinear function of the linear reward in the classical contextual bandit. Two notable instances of this generalized linear reward function are: 1) the multinomial logistic contextual bandit [12], which instantiates the nonlinear function with a multimodule logistic function and 2) logistic contextual bandit [13], which instantiates the nonlinear function

with a logistic function. Different from these variants of the contextual bandit model, our model considers a generalized reward function with latent features. The latent features would introduce a variety of new challenges in designing the learning algorithm. Wang *et al.* [3] also considered a contextual bandit with latent feature, but their model only considers the linear reward function. Allowing a generalized linear reward function introduces some nonlinearity in estimating model parameters, making the problem much more challenging. Krishnamurthy *et al.* [8] generalized the linear reward function to a semi-parametric form, which was further studied in Peng *et al.* [10]. Guan and Jiang [14] generalized the linear reward function to a nonparametric form. For completeness, let us summarize a variety of other notable variants. Zhou and Brunskill [4] proposed a latent contextual bandit model, which associates each user with a latent type. Lamprier *et al.* [5] proposed a latent contextual bandit model that in each round the contextual vector of each action is generated by a normal distribution. Li *et al.* [15] incorporated friendship into the classical contextual bandit. Zeng *et al.* [16] considered a time varying user preference vector. Zhang *et al.* [17] extended the classical contextual bandit to incorporate conversational feedbacks. Lastly, clustering of bandits [18], [19] is also a notable variant.

B. Contextual Bandit Algorithms

There are a number of contextual bandit variants and each variant is associated with a number of algorithms tuned for it. Here, we discuss two lines of contextual bandit algorithms that are closely related to our work: 1) algorithms for contextual bandit with generalized linear reward function and 2) algorithms for contextual bandit with latent features (linear reward). There are two main ideas, i.e., confidence bounds [20] and sampling [21], in the design of algorithms for contextual bandit with generalized linear reward function. There are a variety of confidence bound based algorithms, and to the best of our knowledge, the GLM-UCB proposed by Filippi *et al.* [6] was the first one and performs well in practice [7], [22]. When the generalized linear reward function is in a multinomial logistic form, Oh and Iyengar [12] proposed UCB-MUL algorithm, which refined the GLM-CB algorithm. It is unknown whether GLM-UCB can achieve the optimal rate of regret. Based on the idea of confidence bound, Li *et al.* [7] proposed a SupCB-GLM algorithm and show that it achieves the optimal rate in certain scenarios. Li *et al.* [7], Kveton *et al.* [9]: further improved this algorithm by using a randomized maximum likelihood method to balance exploration vs. exploitation tradeoff. When the generalized linear reward function is in a logistic form, Dumitrascu *et al.* [13] proposed a Thompson sampling algorithm for it. Russo and Van Roy [23] developed a general posterior sampling framework for sequential decision making, which is also applicable to the contextual bandit with generalized linear reward. However, computational issues like how to generate samples from the posterior distributions efficiently are not covered in the framework. The confidence bounds and sampling are also the main ideas in the design of algorithms for contextual bandit with the latent feature. Note that the following algorithms are tuned for the setting of linear reward. Wang *et al.* developed the hLinUCB algorithm [3], which generalized the LinUCB algorithm to deal with latent features. Wang *et al.* further generalized the hLinUCB to obtain the FactorUCB algorithm [24] to the setting that user preference vector $\boldsymbol{\theta}$ has a factorization form $\boldsymbol{\theta} = \boldsymbol{\Theta}\boldsymbol{w}$. Lamprier *et al.* [5] proposed the HiddenLinUCB. The HiddenLinUCB is tuned for the setting that in each round the contextual vector of each action is generated by a normal distribution with certain parameters. To the best of our knowledge, there are no sampling algorithms tuned for contextual bandit with the latent feature. We summarize some sampling algorithms for learning latent features in online collaborative filtering, which shares a certain similarity with the contextual bandit with latent features. Notable examples include [25]–[27]. Different from the above algorithms, we design algorithms for a class of generalized contextual bandit with latent features, and it has a generalized linear reward function tuned for modeling the user behavior in evaluating actions.

III. Sequential Decision Model

We first present a sequential decision framework. Then, we present a generalized contextual bandits framework with latent features and generalized linear reward functions. Finally, we present a design space of sequential decision-making algorithms and a metric to quantify the performance.

A. Sequential Decision Framework

Consider a sequential decision making problem, where a decision maker makes decisions in a finite number of $T \in \mathbb{N}_+$ rounds. Let $\mathcal{A} \subset \mathbb{N}_+$, where $|\mathcal{A}| < \infty$, denote a set of finite actions. For example, the decision maker can be a search engine (or a news recommender system), and $\mathcal{A}$ can be a set of items (or a set of news) in consideration. In round $t \in [T] \triangleq \{1, \ldots, T\}$, the decision maker is given a finite set of choices $\mathcal{A}_t \subseteq \mathcal{A}$, and it needs to select one action $A_t \in \mathcal{A}_t$ from it. Allowing $\mathcal{A}_t$ to change over time can capture important factors such as the lifetime of some items or some news. Once the decision maker takes the action A_t in round t, it receives a reward (or feedback) of $R_t(A_t) \in \mathcal{R}$, where $\mathcal{R} \subset \mathbb{R}$. The reward metric $\mathcal{R}$ is defined by the application. For example, for search engine applications, $\mathcal{R} \triangleq \{0, 1\}$ models whether a user clicks (1) an item or not (0). In news recommendation applications, $\mathcal{R} \triangleq [0, 1]$ quantifies the normalized time that a user spends reading a piece of news. We consider the general case that A_t is a random variable, capturing that the decision maker may randomize the decisions. Furthermore, for each given action A_t, the reward $R_t(A_t)$ is a random variable capturing the uncertainty in user behavior, i.e., clicking items, reading news, etc. We consider a risk neutral decision maker, who aims to maximize the expected cumulative reward $\mathbb{E}[\sum_{t=1}^{T} R_t(A_t)]$.

B. Reward Model

1) Contexts Model: We focus on the case that the decisions in these T rounds are made with the interactions to an agent. In the search engine (or news recommendation) application, the agent represents the whole population of users (or one

single user). Each action $a \in \mathcal{A}$ is associated with a feature vector $\boldsymbol{x}_a \in \mathbb{R}^d$, which summarizes the observed context between the action and the agent, where $d \in \mathbb{N}_+$. For example, $\boldsymbol{x}_a$ can be a summarization of the type of the action (e.g., item, news), or the age of the agent. Let $\boldsymbol{y}_a \in \mathbb{R}^\ell$ denote the unobserved/latent features or contexts, where $\ell \in \mathbb{N}_+$. Let $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\boldsymbol{\vartheta} \in \mathbb{R}^\ell$ denote the agent's preference vector associated with $\boldsymbol{x}_a$ and $\boldsymbol{y}_a$, respectively. We like to emphasize that $\boldsymbol{x}_a, \forall a \in \mathcal{A}$, are known to the decision maker, while the $\boldsymbol{y}_a, \forall a \in \mathcal{A}$, $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$ are unknown to the decision maker.

C. Reward Model

Given $\boldsymbol{x}_a, \boldsymbol{y}_a, \boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$, we consider a two-layer model for the reward (or feedback) generation process. In the first layer, after the decision maker decides an action a in round t, the agent forms its natural evaluation of the action a, modeled by

$$V_t(a) \triangleq \boldsymbol{x}_a^{\mathrm{T}}\boldsymbol{\theta} + \boldsymbol{y}_a^{\mathrm{T}}\boldsymbol{\vartheta} + \epsilon_{t,a} \quad \forall a \in \mathcal{A}_t \tag{1}$$

where $\epsilon_{t,a} \in \mathbb{R}$ denotes a random variable with $\mathbb{E}[\epsilon_{t,a}] = 0$. Higher evaluation $V_t(a)$ implies that the agent is more positive about the action. Note that the evaluation $V_t(a)$ is unknown to the decision maker. The random variable $\epsilon_{t,a}$ captures the noise or uncertainty caused by human factors such as bias in the evaluation. We consider the case that $\epsilon_{t,a}$ across different rounds t and different actions a are independent. Furthermore, for a given action a, the noise $\epsilon_{t,a}$ across different rounds t are identically distributed. Let $f(\epsilon, \sigma_a)$ denote the probability density function of the noise $\epsilon_{t,a}$, where σ_a denotes the standard deviation of $\epsilon_{t,a}$. The standard deviation σ_a is unknown to the decision maker. For simplicity, we denote a collection of all the unknown model parameters as $\boldsymbol{\Psi} \triangleq [\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}]$, where $\boldsymbol{Y} \triangleq [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{|\mathcal{A}|}]$ denotes a matrix of the latent features for all actions, and $\boldsymbol{\sigma} \triangleq [\sigma_1, \ldots, \sigma_{|\mathcal{A}|}]$ denotes a vector of the standard deviations of the agent's evaluation for all the actions.

In the second layer, the agent maps its natural evaluation $V_t(a)$ to the reward metric $\mathcal{R}$. We use the function $g_t : \mathbb{R} \to \mathcal{R}$ to model this mapping process. Namely, the reward under action A_t is $R_t(A_t) = g_t(V_t(A_t))$. The reward $R_t(A_t)$ is known to the decision maker. Recall that the reward $R_t(A_t)$ is the agent's feedback. For example, for search engine applications, a reward (or feedback) is whether an agent clicks (1) an item or not (0). In news recommendation applications, a reward (or feedback) is the degree that an agent likes an item. We assume that g_t is an increasing function to capture that a higher evaluation $V_t(A_t)$ leads to a larger reward or more positive feedback $R_t(A_t)$. Furthermore, the mapping function g_t is allowed to vary over rounds, which captures the evolution of the expertise or leniency in providing feedbacks, etc., of the agent. For example, consider $\mathcal{R} = [0, 1]$, one instance of g_t is the sigmoid function, that is,

$$g_t(V(A_t)) = \frac{1}{1 + \exp(-\gamma_t V(A_t))} \tag{2}$$

where $\gamma_t \in \mathbb{R}_+$ models the leniency of the agent in providing feedbacks. Larger γ_t means that the agent is more lenient. Finally, given the unknown model parameters $\boldsymbol{\Psi} = [\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}]$, the expected reward for action a is defined as

$$\bar{R}_t(a; \boldsymbol{\Psi}) \triangleq \mathbb{E}[R_t(a)|\boldsymbol{\Psi}] = \int_{\mathbb{R}} f(\epsilon, \sigma_a) g_t\big(\boldsymbol{x}_a^{\mathrm{T}}\boldsymbol{\theta} + \boldsymbol{y}_a^{\mathrm{T}}\boldsymbol{\vartheta} + \epsilon\big) d\epsilon.$$

Namely, the expected reward $\bar{R}_t(a)$ is in a generalized linear form and contains latent features. Note that $\bar{R}_t(a)$ is unknown to the decision maker because $\sigma_a, \boldsymbol{\theta}$, etc., are unknown.

D. Sequential Decision Making Algorithms

1) Design Space of Sequential Decision Making Algorithms: Recall that in each round t after the decision is made, the action A_t, the reward $R_t(A_t)$, and the observed context $\boldsymbol{x}_t$ are known to the decision maker. Let $\mathcal{H}_t$ denote the decision history up to decision round t, which is defined as

$$\mathcal{H}_t \triangleq \big\{\big[A_1, \boldsymbol{x}_{A_1}, R_1(A_1)\big], \ldots, \big[A_t, \boldsymbol{x}_{A_t}, R_t(A_t)\big]\big\}.$$

In decision round t, the decision maker knows only the decision history up to round $t-1$. Thus, we consider a design space of history-dependent sequential decision making algorithms, which prescribe each decision history $\mathcal{H}_{t-1}$ to a probability distribution over the actions $\mathcal{A}_t$. We use a general probability distribution $\mathcal{D}(\mathcal{H}_{t-1})$ over $\mathcal{A}_t$ to represent a general history-dependent sequential decision making algorithm. Namely, the action A_t is generated from the distribution $\mathcal{D}(\mathcal{H}_{t-1})$, $A_t \sim \mathcal{D}(\mathcal{H}_{t-1})$. This type of decision algorithm captures that the decision maker may randomize its actions. The deterministic case is a special case that the distribution $\mathcal{D}(\mathcal{H}_t)$ concentrates on one action with zero variance.

2) Performance Metric: Recall that we consider a risk neutral decision maker. Thus, we define the following regret function to quantify the performance of a history-dependent sequential decision making algorithm with probabilistic representation $\mathcal{D}$

$$R_T(\mathcal{D}; \boldsymbol{\Psi}) \triangleq \sum_{t=1}^{T} \max_{a \in \mathcal{A}_t} \bar{R}_t(a; \boldsymbol{\Psi}) - \mathbb{E}\left[\sum_{t=1}^{T} R_t(A_t) \Big| \boldsymbol{\Psi}, A_t \sim \mathcal{D}(\mathcal{H}_{t-1})\right] \tag{3}$$

where $\boldsymbol{\Psi} = [\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}]$ denotes a collection of all unknown model parameters.

The decision maker may have some prior knowledge on the latent features $\boldsymbol{Y}$, preference vector $(\boldsymbol{\theta}, \boldsymbol{\vartheta})$ and the standard deviation $\boldsymbol{\sigma}$ of the agent's valuations for each action. Formally, we model the prior knowledge as the prior distributions over these unknown parameters, denoted by $p(\boldsymbol{y}_a), p(\sigma_a), \forall a \in \mathcal{A}$ and $p(\boldsymbol{\theta}, \boldsymbol{\vartheta})$. We focus on the case that the latent feature vectors $\boldsymbol{y}_a, \forall a \in \mathcal{A}$, preference vector $(\boldsymbol{\theta}, \boldsymbol{\vartheta})$, and standard deviation $\sigma_a, \forall a \in \mathcal{A}$ are independently generated from their own prior distributions, i.e., $p(\boldsymbol{\Psi}) = p(\boldsymbol{\theta}, \boldsymbol{\vartheta}) \prod_{a \in \mathcal{A}} p(\boldsymbol{y}_a) p(\sigma_a)$, where $p(\boldsymbol{\Psi})$ denotes the joint prior distribution over the unknown model parameters $\boldsymbol{\Psi} = [\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}]$. We formally define the Bayesian regret to quantify the performance of a history-dependent sequential decision-making algorithm with prior knowledge as

$$R_T^{\mathrm{Bay}}(\mathcal{D}) \triangleq \int R_T(\mathcal{D}; \boldsymbol{\Psi}) p(\boldsymbol{\Psi}) d\boldsymbol{\Psi}.$$

Algorithm 1 GCL-PS (Generalized Contextual Bandit With Latent Feature via Posterior Sampling)

1: Initialize $\mathcal{H}_0 = \emptyset$
2: Notations: the samples of unknown parameters in round t

$$\boldsymbol{Y}_t \triangleq \left[\boldsymbol{y}_{1,t}, \ldots, \boldsymbol{y}_{|\mathcal{A}|,t}\right], \boldsymbol{\sigma}_t \triangleq \left[\sigma_{1,t}, \ldots, \sigma_{|\mathcal{A}|,t}\right]$$
$$\boldsymbol{\theta}_t \triangleq \left[\theta_{1,t}, \ldots, \theta_{d,t}\right], \boldsymbol{\vartheta}_t \triangleq \left[\vartheta_{1,t}, \ldots, \vartheta_{\ell,t}\right],$$
$$\boldsymbol{\Psi}_t \triangleq [\boldsymbol{Y}_t, \boldsymbol{\theta}_t, \boldsymbol{\vartheta}_t, \boldsymbol{\sigma}_t].$$

3: **for** $t = 1, 2, 3, \ldots, T$ **do**
4: $\quad \boldsymbol{\Psi}_t \sim p(\boldsymbol{\Psi}|\mathcal{H}_{t-1})$ derived in (4)
5: $\quad$ Select action by $A_t \leftarrow \arg\max_{a \in \mathcal{A}_t} \bar{R}_t(a; \boldsymbol{\Psi}_t)$
6: $\quad$ Observe reward $R_t(A_t)$
7: $\quad$ Update history $\mathcal{H}_t \leftarrow \mathcal{H}_{t-1} \cup \{[A_t, \boldsymbol{x}_{A_t}, R_t(A_t)]\}$
8: **end for**

The objective of the decision maker is to design a history-dependent sequential decision-making algorithm so as to attain a Bayesian regret as small as possible.

IV. Algorithms and Regret Analysis

We first design a GCL-PS algorithm for the generalized contextual bandit framework formulated in Section III. We prove both upper and lower Bayesian bounds for GCL-PS and also show that it may have a high computational complexity. Then, we design GCL-PSMC algorithm which improves the computational efficiency of GCL-PS via MCMC.

A. GCL-PS Algorithm and Computational Complexity

We first present the design of the GCL-PS algorithm. Then we prove the sublinear upper bound for the Bayesian regret of the GCL-PS algorithm, as well as use an example to illustrate high computational complexity of the GCL-PS algorithm.

1) Algorithm Design: Given a decision history $\mathcal{H}_t$ up to round t, let $p(\boldsymbol{\Psi}|\mathcal{H}_t)$ denote the posterior distribution of the unknown model parameters $\boldsymbol{\Psi} = [\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}]$. In the following lemma, we derive a closed-form formula for this posterior distribution.

Lemma 1: Given the decision history $\mathcal{H}_t$ up to round t, the posterior distribution $p(\boldsymbol{\Psi}|\mathcal{H}_t)$ can be derived as

$$p(\boldsymbol{\Psi}|\mathcal{H}_t) = \frac{1}{Z}\left[p(\boldsymbol{\theta}, \boldsymbol{\vartheta}) \prod_{a \in \mathcal{A}} p\left(\boldsymbol{y}_a\right) p(\sigma_a)\right] \times \left[\prod_{\tau=1}^{t-1} \prod_{a \in \mathcal{A}_\tau} \left[f\left(g_\tau^{-1}(R_\tau(a)) - \boldsymbol{x}_a^{\mathrm{T}}\boldsymbol{\theta} - \boldsymbol{y}_a^{\mathrm{T}}\boldsymbol{\vartheta}, \sigma_a\right)\right]^{\mathbb{1}\{A_\tau = a\}}\right] \quad (4)$$

where Z denotes the normalizing factor and it is independent of the unknown model parameters $\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}$ and $\boldsymbol{\sigma}$.

a) All proofs are presented in supplementary file: Lemma 1 states a closed-form formula for us to generate samples from the posterior distribution $p(\boldsymbol{\Psi}|\mathcal{H}_t)$. Based on Lemma 1, Algorithm 1 outlines a posterior sampling algorithm to solve the generalized contextual bandit learning problem formulated in Section III. The idea of Algorithm 1 is as follows. In each round t, we first generate samples of the unknown model parameters using the posterior distribution derived in (4). Then we input these samples into our model to estimate the expected reward of each action, based on which we select the action with the largest estimated reward. After the decision maker takes the selected action to the agent, it will observe a reward from the agent. Finally, the decision maker updates the decision history to include the observed reward. After this update, the decision goes into round $t + 1$.

2) Algorithm Analysis: The next theorem states an upper bound for the Bayesian regret of Algorithm 1.

Theorem 1: Suppose $\epsilon_{t,a}$ is sub-Gaussian with variance proxy $\xi_{t,a}$, i.e., $\mathbb{E}[\exp(c\epsilon_{t,a})] \leq \exp(c^2\xi_{t,a}^2/2), \forall c \in \mathbb{R}$. Suppose the function g_t is $\zeta_t \in \mathbb{R}_+$ Lipschitz and the dimension of latent feature is at least 1, i.e., $\ell \geq 1$. Suppose the observed features are bounded, i.e., $\|\boldsymbol{x}_a\| \leq L, \forall a \in \mathcal{A}$. Define notation $\xi_{\max} \triangleq \max_{a \in \mathcal{A}, \tau \leq T} \xi_{\tau,a}$ and define notation $\Delta(\mathcal{R}) \triangleq \max \mathcal{R} - \min \mathcal{R}$. The Bayesian regret upper bound of Algorithm 1 can be derived as

$$R_T^{\mathrm{Bay}}(\mathcal{D}_{\mathrm{GCL-PS}}) \leq \sqrt{2T(d+|\mathcal{A}|)\log(1+T(L+1)/(d+|\mathcal{A}|))} \left[2\max_{\tau \leq T} \zeta_\tau \left(\xi_{\max}\sqrt{(d+|\mathcal{A}|)\log\left(T+T^2(L+1)\right)}\right) + \mathbb{E}_{\boldsymbol{\Psi} \sim p(\boldsymbol{\Psi})}\left[\sqrt{\sum_{i=1}^{d}\theta_i^2 + \sum_{a \in \mathcal{A}}\left(\boldsymbol{y}_a^T\boldsymbol{\vartheta}\right)^2}\right] + \Delta(\mathcal{R})\right]$$

where $\mathcal{D}_{\mathrm{GCL-PS}}$ denotes Algorithm 1.

Theorem 1 states that the attractive property of the Bayesian regret of Algorithm 1 is *sublinear* in the number of rounds T with an order of $(T)^{1/2} \ln T$. This implies that the average regret of Algorithm 1 per round converges to zero as the number of rounds T goes to infinity. Furthermore, the regret upper bound reveals the impact of model parameters on the learning speed. In particular, the regret upper bound increases linearly in ζ_t implying that the learning speed increases in the smoothness of g_t. The regret upper bound increases linearly in $\xi_{\max}$, i.e., the learning speed decreases as the tail of the distribution of noise $\epsilon_{t,a}$ becomes heavier. Lastly, the regret upper bound is nearly linear in $d + |\mathcal{A}|$. This implies that when the number of arms is large, the GCL-PS may suffer a slow learning speed. To answer whether we can eliminate this dependency on $|\mathcal{A}|$, we also derive regret lower bound in the following theorem.

Theorem 2: For any algorithm $\mathcal{D}$, there is an instance of the generalized contextual bandits with the latent feature model such that the Bayesian regret satisfies $R_T^{\mathrm{Bay}}(\mathcal{D}) \geq \Omega((T|\mathcal{A}|)^{1/2})$.

Theorem 2 states that for any algorithm $\mathcal{D}$, there is an instance of the model such that the regret is at least in the order of $(T|\mathcal{A}|)^{1/2}$. In other words, in general, we cannot design a decision algorithm such that its regret is independent of $|\mathcal{A}|$. Furthermore, when the number of arms $|\mathcal{A}|$ is large, no algorithm can eliminate the risk of slow learning speed.

Although Algorithm 1 has a nice Bayesian regret upper bound, it is computationally expensive to implement in general. This is because step 4 of Algorithm 1, i.e., generating samples from the posterior distribution $p(\boldsymbol{\Psi}|\mathcal{H}_{t-1})$ derived in (4), is computationally expensive in general. To illustrate, consider the following simple case. Assume that the noise $\epsilon_{t,a}$

follows a Gaussian distribution, i.e., $f(\epsilon, \sigma_a)$ is the probability density function of a Gaussian distribution $\mathcal{N}(0, \sigma_a^2)$. The prior distribution for $\boldsymbol{y}_a$ and $(\boldsymbol{\theta}, \boldsymbol{\vartheta})$ are an ℓ-dimensional and a $(d+\ell)$-dimensional Gaussian distribution, respectively, i.e., $p(\boldsymbol{y}_a)$ and $p(\boldsymbol{\theta}, \boldsymbol{\vartheta})$ are the probability density function of $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{\ell\times\ell})$ and $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{(d+\ell)\times(d+\ell)})$, respectively. The prior distribution for σ_a is an inverse gamma distribution, i.e., $p(\sigma_a)$ is the probability density function of Inv-Gamma(1, 1). Plugging these probability density functions into (4), one can obtain a refined posterior distribution as

$$p(\boldsymbol{\Psi}|\mathcal{H}_t) = \left[\prod_{\tau=1}^{t-1} \frac{1}{\sqrt{2\pi\sigma_{A_\tau}^2}} \exp\left(-\frac{g_\tau^{-1}(R_\tau(A_\tau)) - \boldsymbol{x}_{A_\tau}^{\mathrm{T}}\boldsymbol{\theta} - \boldsymbol{y}_{A_\tau}^{\mathrm{T}}\boldsymbol{\vartheta}}{2\sigma_{A_\tau}^2}\right)\right] \times \left[\prod_{a\in\mathcal{A}} \frac{1}{(2\pi)^{\ell/2}} \exp\left(-\frac{\boldsymbol{y}_a^T\boldsymbol{y}_a}{2}\right) \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma_a}\right)^2 \exp\left(-\frac{1}{\sigma_a}\right)\right] \times \left[\frac{1}{(2\pi)^{(d+\ell)/2}} \exp\left(-\frac{\boldsymbol{\theta}^T\boldsymbol{\theta} + \boldsymbol{\vartheta}^T\boldsymbol{\vartheta}}{2}\right)\right] \frac{1}{Z}. \quad (5)$$

To the best of our knowledge, there is no computationally efficient algorithm to generate exact samples from this posterior distribution derived in (5). To overcome this issue, we next develop an MCMC algorithm to generate approximate samples from the general posterior distribution, improving the computational efficiency at a slight sacrifice of the sequential decision accuracy.

Algorithm 2 GCL-PSMC (GCL-PS With MCMC)

1: Initialize: $\mathcal{H}_0 = \emptyset$; $\boldsymbol{y}_{a,0} \sim p(\boldsymbol{y}_a), \forall a \in \mathcal{A}$; $\sigma_{a,0} \sim p(\sigma_a), \forall a \in \mathcal{A}$; $(\boldsymbol{\theta}_0, \boldsymbol{\vartheta}_0) \sim p(\boldsymbol{\theta}, \boldsymbol{\vartheta})$;
2: Notation: $\boldsymbol{\Psi}_t$ denotes the sample of the unknown model parameters generated in round t as defined in Algorithm 1. We define the samples generated by the n-th iteration of the MCMC as
$$\boldsymbol{Y}^{(n)} \triangleq \left[\boldsymbol{y}_1^{(n)}, \ldots, \boldsymbol{y}_{|\mathcal{A}|}^{(n)}\right], \boldsymbol{\sigma}^{(n)} \triangleq \left[\sigma_1^{(n)}, \ldots, \sigma_{|\mathcal{A}|}^{(n)}\right]$$
$$\boldsymbol{\theta}^{(n)} \triangleq \left[\theta_1^{(n)}, \ldots, \theta_d^{(n)}\right], \boldsymbol{\vartheta}^{(n)} \triangleq \left[\vartheta_1^{(n)}, \ldots, \vartheta_\ell^{(n)}\right].$$
3: $\boldsymbol{\Psi}_0 \leftarrow [\boldsymbol{Y}_0, \boldsymbol{\theta}_0, \boldsymbol{\vartheta}_0, \boldsymbol{\sigma}_0]$
4: **for** $t = 1, 2, 3, \ldots, T$ **do**
5: Initialize MCMC: $[\boldsymbol{Y}^{(0)}, \boldsymbol{\theta}^{(0)}, \boldsymbol{\vartheta}^{(0)}, \boldsymbol{\sigma}^{(0)}] \leftarrow \boldsymbol{\Psi}_{t-1}$
6: **for** $n = 1, 2, 3, \ldots, N$ **do**
7: $[\boldsymbol{\theta}^{(n)}, \boldsymbol{\vartheta}^{(n)}] \sim p(\boldsymbol{\theta}, \boldsymbol{\vartheta} | \boldsymbol{Y} = \boldsymbol{Y}^{(n-1)}, \boldsymbol{\sigma} = \boldsymbol{\sigma}^{(n-1)}, \mathcal{H}_{t-1})$
8: $\sigma_a^{(n)} \sim p(\sigma_a | \boldsymbol{Y} = \boldsymbol{Y}^{(n-1)}, \boldsymbol{\theta} = \boldsymbol{\theta}^{(n)}, \boldsymbol{\vartheta} = \boldsymbol{\vartheta}^{(n)}, \mathcal{H}_{t-1}), \forall a$
9: $\boldsymbol{y}_a^{(n)} \sim p(\boldsymbol{y}_a | \boldsymbol{\sigma} = \boldsymbol{\sigma}^{(n)}, \boldsymbol{\theta} = \boldsymbol{\theta}^{(n)}, \boldsymbol{\vartheta} = \boldsymbol{\vartheta}^{(n)}, \mathcal{H}_{t-1}), \forall a$
10: **end for**
11: $\boldsymbol{\Psi}_t \leftarrow (\boldsymbol{Y}^{(N)}, \boldsymbol{\theta}^{(N)}, \boldsymbol{\vartheta}^{(N)}, \boldsymbol{\sigma}^{(N)})$
12: Select action by $A_t \leftarrow \arg\max_{a\in\mathcal{A}_t} \bar{R}(a; \boldsymbol{\Psi}_t)$
13: Observe reward $R_t(A_t)$
14: Update history $\mathcal{H}_t \leftarrow \mathcal{H}_{t-1} \cup [A_t, \boldsymbol{x}_{A_t}, R_t(A_t)]$
15: **end for**

B. MCMC to Improve Efficiency

We first present a general *GCL-PSMC* algorithm, which uses a MCMC algorithm to generate approximate samples from the posterior distribution $p(\boldsymbol{\Psi}|\mathcal{H}_t)$. We not only derive a Bayesian regret upper bound for the *GCL-PSMC* algorithm, but also reveal insights on how the approximate samples may influence the regret. Finally, we present an instance of the *GCL-PSMC* algorithm with Gaussian distributions, in which sampling efficiency can be further improved.

1) General GCL-PSMC Algorithm: Algorithm 2 outlines the design of the *GCL-PSMC* algorithm. The difference between Algorithm 2 and Algorithm 1 is that it uses an MCMC algorithm, i.e., a three-stage Gibbs sampler, to generate approximate samples from the posterior distribution $p(\boldsymbol{\Psi}|\mathcal{H}_t)$. The idea of the three-stage Gibbs sampler is as follows. The unknown model parameters that we aim to sample can be categorized into three groups, i.e., the latent feature $\boldsymbol{Y}$, preference vector $(\boldsymbol{\theta}, \boldsymbol{\vartheta})$ and the standard deviation $\boldsymbol{\sigma}$ of the agent's valuations associated with each action. Furthermore, (4) implies that the conditional posterior distribution of $\boldsymbol{y}_a$ (given $\boldsymbol{\theta}$, $\boldsymbol{\vartheta}$ and $\boldsymbol{\sigma}$) across different actions a are independent. This independence across actions also holds for the conditional posterior distribution of σ_a (given $\boldsymbol{\theta}$, $\boldsymbol{\vartheta}$ and $\boldsymbol{Y}$). Lastly, the preference vector $(\boldsymbol{\theta}, \boldsymbol{\vartheta})$ appears in a linear form given $\boldsymbol{Y}$ and $\boldsymbol{\sigma}$. Based on these observations, we design a three-stage Gibbs sampler (i.e., step 5–10 of Algorithm 2). Each iteration of the three-stage Gibbs sampler consists of the following: 1) in the first stage, we sample the preference vector $(\boldsymbol{\theta}, \boldsymbol{\vartheta})$ using its condition posterior distribution given the latest samples of $\boldsymbol{Y}$ and $\boldsymbol{\sigma}$; 2) in the second stage, we sample the standard deviation $\boldsymbol{\sigma}$ using its condition posterior distribution given the latest samples of $\boldsymbol{\theta}$, $\boldsymbol{\vartheta}$ and $\boldsymbol{Y}$; and 3) in the third stage, we sample the latent feature $\boldsymbol{Y}$ using its condition posterior distribution given the latest samples of $\boldsymbol{\theta}$, $\boldsymbol{\vartheta}$ and $\boldsymbol{\sigma}$. We like to point out that we use the samples of $\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}$ from the previous decision round as the initial point of this three-stage Gibbs sampler (i.e., step 5 of Algorithm 2). The intuition is that in each decision round, we only use one reward to update the posterior distribution. This means that the posterior distributions in two consecutive rounds do not differ a lot. Thus, the samples of $\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}$ in last round are not far from the stationary distribution (i.e., posterior distribution) in current round.

The following theorem states that under mild assumptions on the model, the three-stage Gibbs sampler described in the step 5–10 of Algorithm 2 converges to the stationary distribution, which is the same as $p(\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}|\mathcal{H}_t)$.

Theorem 3: Suppose $f(\epsilon, \sigma_a)$ is continuous and positive on $\mathbb{R}$, i.e., $f(\epsilon, \sigma_a) > 0, \forall \epsilon \in \mathbb{R}$. Suppose g_t is continuously increasing in $\mathbb{R}$ and it is an injective function from $\mathbb{R}$ to $\mathcal{R}$. Suppose the density functions of the prior distributions have supports equal the domain of the model parameters. Then, the three-stage Gibbs sampler described in step 5–10 of Algorithm 2, converges to $p(\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}|\mathcal{H}_t)$ as N goes to infinity.

Theorem 3 states that the three-stage sampler in Algorithm 2 can generate samples following a distribution arbitrarily close to the posterior distribution $p(\boldsymbol{\Psi}|\mathcal{H}_t)$ when the number of iterations N is sufficiently large. Intuitively, the condition in Theorem 3 identifies some well-behaved distributions in the sense that they are continuous and have positive values on

the domain of the unknown model parameters. The Gaussian example in Section IV-A satisfies these conditions.

Theorem 4: Consider the same conditions as Theorem 1. Let $p_t^{(N)}(\cdot)$ denote the probability distribution of the samples generated by the three-stage sampler in Algorithm 2 with N iterations. If $p_t^{(N)}(\cdot)$ satisfies

$$\|p_t^{(N)}(\cdot) - p(\cdot|\mathcal{H}_{t-1})\|_{TV} \leq \eta/\sqrt{t}$$

where $\|\cdot\|_{TV}$ denotes the total variation distance and $p(\cdot|\mathcal{H}_{t-1})$ denotes the posterior distribution, Algorithm 2 has a Bayesian regret of

$$R_T^{\text{Bay}}(\mathcal{D}_{\text{GCL-PSMC}}) \leq R_T^{\text{Bay}}(\mathcal{D}_{\text{GCL-PS}}) + 2\max_{r\in\mathcal{R}}|r|\sqrt{T}\eta$$

where $\mathcal{D}_{\text{GCL}-\text{PSMC}}$ represents Algorithm 2.

Theorem 4 states that the *GCL-PSMC* can achieve the same order of sublinear Bayesian regret in T, if the samples generated from the three-stage Gibbs sampler of Algorithm 2 approximate the posterior distribution well. Furthermore, it provides a connection between the Bayesian regret and the "accuracy" of the approximate samples. This helps us to attain different tradeoffs between regret and computational cost in generating accurate approximate samples. More importantly, this tradeoff guides us to determine the stopping condition for the MCMC algorithm. The following corollary states the number rounds N for simulating the Markov chain such that the Bayesian regret upper bound of GCL-PSMC is sublinear.

Corollary 1: Suppose the Markov chain associated with Algorithm 2 is uniform ergodic, i.e., $\|p_t^{(N)}(\cdot) - p(\cdot|\mathcal{H}_{t-1})\| \leq c\rho^N$, where $c > 0$ and $\rho \in (0, 1)$. By setting $N = -(0.5\ln t + \ln c)/\ln\rho$ in decision round t, we have

$$R_T^{\text{Bay}}(\mathcal{D}_{\text{GCL-PSMC}}) \leq R_T^{\text{Bay}}(\mathcal{D}_{\text{GCL-PS}}) + \max_{r\in\mathcal{R}}|r|\sqrt{T}.$$

Corollary 1 states that when the Markov chain associated with Algorithm 2 is uniform ergodic, the number rounds N for simulating the Markov chain is in an order of $\ln t$. When the uniform ergodicity of the Markov chain is difficult to establish, one can use Theorem 4 to select appropriate η, and then for each selected η use convergence diagnose methods [28] to determine the stopping condition of simulating the Markov chain. In the following, we use an instance of the GCL-PSMC algorithm to show that it can be implemented efficiently.

2) Instance of the GCL-PSMC With Gaussians: We consider the following instance of our model with Gaussians.

Instance 1: Consider the instance of the model with: 1) the prior of the latent feature and preference vector follow Gaussian distributions, i.e., $p(\boldsymbol{y}_a)$ is the density function of $\mathcal{N}(\boldsymbol{v_a}, \boldsymbol{\Lambda}_a)$ and $p(\boldsymbol{\theta}, \boldsymbol{\vartheta})$ is the density function of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ respectively; 2) the prior of the noise follows a Gaussian distribution, i.e., $f(\epsilon, \sigma_a)$ is the density function of $\mathcal{N}(0, \sigma_a^2)$; and 3) the prior of the variance σ_a^2 follows an inverse Gamma distribution, i.e., $p(\sigma_a^2)$ is the density function of Inv-Gamma(α, β).

Given Instance 1, we formally derive the conditional posterior distribution in Algorithm 2 in the following lemma.

Lemma 2: Consider Instance 1. The $p(\boldsymbol{y}_a|\boldsymbol{\sigma}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \mathcal{H}_t)$ follows $\mathcal{N}(\boldsymbol{v}_{a,t}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}), \boldsymbol{\Lambda}_{a,t}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}))$, where

$$\begin{aligned}\boldsymbol{\Lambda}_{a,t}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}) &= \left(\boldsymbol{\Lambda}_a^{-1} + n_{a,t-1}\sigma_a^{-2}\boldsymbol{\vartheta}\boldsymbol{\vartheta}^T\right)^{-1}\\ \boldsymbol{v}_{a,t}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma}) &= \boldsymbol{\Lambda}_{a,t}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\sigma})\Big(\boldsymbol{\Lambda}_a^{-1}\boldsymbol{v}_a\\ &\quad + \boldsymbol{\vartheta}\sigma_a^{-2}\left(\sum_{\tau=1}^{t-1}\mathbb{1}_{\{A_\tau=a\}}g_\tau^{-1}(R_\tau(A_\tau)) - n_{a,t-1}\boldsymbol{\theta}^T\boldsymbol{x}_a\right)\Big).\end{aligned}$$

The $p(\boldsymbol{\theta}, \boldsymbol{\vartheta}|\boldsymbol{Y}, \boldsymbol{\sigma}, \mathcal{H}_t)$ follows $\mathcal{N}(\boldsymbol{\mu}_t(\boldsymbol{Y}, \boldsymbol{\sigma}), \boldsymbol{\Sigma}_t(\boldsymbol{Y}, \boldsymbol{\sigma}))$, where

$$\begin{aligned}\boldsymbol{\Sigma}_t(\boldsymbol{Y}, \boldsymbol{\sigma}) &= \left(\boldsymbol{\Sigma}^{-1} + \sum_{a\in\mathcal{A}} n_{a,t-1}\sigma_a^{-2}\left[\boldsymbol{x}_a^T, \boldsymbol{y}_a^T\right]^T\left[\boldsymbol{x}_a^T, \boldsymbol{y}_a^T\right]\right)^{-1}\\ \boldsymbol{\mu}_t(\boldsymbol{Y}, \boldsymbol{\sigma}) &= \boldsymbol{\Sigma}_t(\boldsymbol{Y}, \boldsymbol{\sigma})\Big(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\\ &\quad + \sum_{a\in\mathcal{A}}\left[\boldsymbol{x}_a^T, \boldsymbol{y}_a^T\right]^T\sigma_a^{-2}\sum_{\tau=1}^{t-1}\mathbb{1}_{\{A_\tau=a\}}g_\tau^{-1}(R_\tau(A_\tau))\Big).\end{aligned}$$

The $p(\sigma_a^2|\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}, \mathcal{H}_t)$ follows Inv-Gamma$(\alpha_t, \beta_t(\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}))$:

$$\begin{aligned}\beta_t(\boldsymbol{Y},\boldsymbol{\theta},\boldsymbol{\vartheta}) &= \beta + \frac{\sum_{\tau=1}^{t-1}\mathbb{1}_{\{A_\tau=a\}}\left[g_\tau^{-1}(R_\tau(A_\tau)) - \boldsymbol{x}_a^T\boldsymbol{\theta} - \boldsymbol{y}_a^T\boldsymbol{\vartheta}\right]^2}{2}\\ \alpha_t &= \alpha + n_{a,t-1}/2\end{aligned}$$

where $n_{a,t-1}$ denotes the number of rounds that action a was selected in the first $t-1$ rounds.

Lemma 2 provides the closed-form formula for the conditional posterior distributions in Algorithm 2 under Instance 1. These conditional posterior distributions follow either Gaussian distributions or inverse Gamma distributions. More importantly, these conditional posterior distribution can be efficiently sampled. Thus, we can efficiently implement Algorithm 2 under Instance 1.

V. Experiments on Synthetic Data

We conduct experiments on synthetic data to evaluate the performance of GCL-PSMC. We not only compare GCL-PSMC with five baselines but also study the impact of various model parameters on the performance of GCL-PSMC.

A. Experiment Settings

1) Synthetic Data: We set the total number of decision rounds to $T = 2000$. We consider a total number of $|\mathcal{A}| = 40$ actions and in each round they are all presented to the decision maker, i.e., $\mathcal{A}_t = \mathcal{A}, \forall t \in [T]$. We use a normal distribution with variance $\sigma_a^2 = 1$ to simulate the noise of the agent's evaluation described in (1), i.e., $f(\epsilon, \sigma_a)$ is the density function of $\mathcal{N}(\epsilon, 1)$. We consider one typical instance of the mapping function $g_t(V) = 1/[1 + \exp(-V)], \forall t \in [T]$, which is the well-known sigmoid function. Unless we explicitly vary the dimension of features, we consider the default choice, i.e., dimensions for the observed and latent feature are $d = 5$ and $\ell = 5$ respectively. Without loss of generality, we consider that the observed feature and the associated preference vector lie in $\boldsymbol{x}_a \in [0, 1]^d, \boldsymbol{\theta} \in [0, 1]^d$; the latent feature and the associated preference vector lie in $\boldsymbol{y}_a \in [0, L]^\ell, \boldsymbol{\vartheta} \in [0, L]^\ell$, where $L \in \mathbb{R}_+$. Unless we vary L explicitly, we set $L = 2$

by default, representing that the latent feature has a slightly stronger effect on the reward than the observed feature. In the regret computation part, we will show how we select the feature vector $\boldsymbol{x}_a$, $\boldsymbol{y}_a$ and the preference vector $\boldsymbol{\theta}$, $\boldsymbol{\vartheta}$.

2) Baselines and Metrics: To the best of our knowledge, this is the first article that studies the generalized contextual bandit with latent features. We aim to reveal fundamental understandings on the impact of the latent feature and nonlinearity of the reward function on the regret. Thus, we compare our algorithm with the following five algorithms.

1) *GCL-LIN:* It is an adaption of the LinUCB [2] to our model. GCL-LIN uses the same method as LinUCB to learn the preference parameter $\boldsymbol{\theta}$ using $\boldsymbol{x}_{A_\tau}$ as feature vector and $g_\tau^{-1}(R_\tau(A_\tau))$ as reward. For each estimated preference parameter $\boldsymbol{\theta}_t$, GCL-LIN estimates the average reward for each arm a as $\mathbb{E}[g_t(\boldsymbol{x}_a^T\boldsymbol{\theta}_t+\epsilon_{t,a})]$ (the method of calculating this expectation will be specified later). GCL-LIN uses the same penalty term as the LinUCB to do exploration. The GCL-LIN algorithm only uses the observed feature to select actions.
2) *GLM-TSL [9]:* It is one of the latest algorithms for contextual bandit with generalized linear model. Different from the UCB method or Thompson sampling method, it uses the randomized maximum likelihood method to balance exploration vs. exploitation tradeoff. Note that it does not consider latent features. We implement this algorithm only use the observed features and the known function g_t.
3) *UCB-GLM [7]:* it is one of the latest UCB based algorithms with provable optimality guarantee in the sense that it matches the minimax lower bound. Note that it does not consider latent features. We implement this algorithm by only using the observed features and the known function g_t. Comparison to these algorithms reveals the impact of the latent feature on regret.
4) *hLinUCB [3]:* It is one of the representative confidence bound-based algorithms for contextual bandit (linear reward function) with latent features. We set its parameters according to this article [3]. This algorithm assumes a linear reward function in selecting actions. Comparison to this algorithm reveals the impact of nonlinearity of the reward function on the regret.
5) *TS-ICF [27]:* It is a Thompson sampling algorithm, which is shown to have a good performance in learning the latent features in matrix factorization scenarios. Note that the contextual bandit is like conducting the matrix factorization for one user. Based on this, we tune this algorithm to our setting. Note that this algorithm considers a linear reward function. Comparison to this algorithm further reveals the impact of nonlinearity of the reward function on the regret of sampling-based sequential decision-making algorithms.

For all the above algorithms in comparison, we focus on the setting that the variance of the agent's evaluation σ_a^2 is known to the algorithm. The reasons are as follows. The regret of the confidence-based algorithms like GCL-LIN and hLinUCB, is sensitive to prior knowledge on the variance σ_a^2 (e.g., upper bound of it). Mis-specifying a prior knowledge on the variance would lead to over exploration (large regret) or under exploration (diverge). Thus, one way to conduct fair comparison is to let all these algorithms in comparison know the variance. Then one can computer the mean $\mathbb{E}[g_t(\boldsymbol{x}_a^T\boldsymbol{\theta}_t+\epsilon_{t,a})]$ for GCL-LIN via integration or Monte Carlo simulation. We further set the prior distribution for the GCL-PSMC as $p(\boldsymbol{y}_a)=\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$, $p(\boldsymbol{\theta},\boldsymbol{\vartheta})=\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$. As the variance is known, then in the sampling based algorithms like GCL-PSMC, there is no need to sample the variance. We just set it to be the given variance. We aim to compare the regret of each algorithm described above as defined in (3).

3) Regret Computation: We aim to compare the regret of each algorithm as defined in (3). Equation (3) states that the regret varies with the parameters $\boldsymbol{x}_a$, $\boldsymbol{y}_a$, $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$ of the model. In practice, different applications may have different parameters. Note that in the synthetic data, we select a parameter space of $(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{\theta},\boldsymbol{\vartheta})\in\Theta\triangleq[0,1]^{d\times|\mathcal{A}|}\times[0,L]^{\ell\times|\mathcal{A}|}\times[0,1]^d\times[0,L]^\ell$, where $\boldsymbol{X}\triangleq[\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{|\mathcal{A}|}]$ and $\boldsymbol{Y}\triangleq[\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{|\mathcal{A}|}]$. Instead of selecting some instances of these parameters to study, we aim to compute the average of the cumulative regret over the whole parameter space

$$\bar{R}_T(\mathcal{D})\triangleq\int_\Theta R_T(\mathcal{D};\boldsymbol{\Psi})d\mu(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{\theta},\boldsymbol{\vartheta}) \tag{6}$$

where $\boldsymbol{\Psi}=[\boldsymbol{Y},\boldsymbol{\theta},\boldsymbol{\vartheta},\boldsymbol{\sigma}]$, $\boldsymbol{\sigma}=\boldsymbol{1}$ as set in the synthetic data, and $\mu(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{\theta},\boldsymbol{\vartheta})$ denotes a Lebesgue measure (i.e., the "weight" associated with each selection of parameter $(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{\theta},\boldsymbol{\vartheta})$ is equal) over the parameter space Θ. Recall that $\mathcal{D}$ represents a probabilistic representation of an algorithm. As implied by a result in [23], the average cumulative regret $\bar{R}_T(\mathcal{D})$ connects to the Bayesian regret as follows:

$$\bar{R}_T(\mathcal{D})\leq O\left(\frac{R_T^{\text{Bay}}(\mathcal{D})}{\max_{(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{\theta},\boldsymbol{\vartheta})\in\Theta}\left[p(\boldsymbol{\theta},\boldsymbol{\vartheta})\prod_{a\in\mathcal{A}}p(\boldsymbol{y}_a)\right]}\right). \tag{7}$$

Inequality (7) implies that if the Bayesian regret $R_T^{\text{Bay}}(\mathcal{D})$ is sublinear, the average cumulative regret is also sublinear. Note that in the synthetic data we select a parameter space of $\boldsymbol{x}_a\in[0,1]^d$, $\boldsymbol{\theta}\in[0,1]^d$, $\boldsymbol{y}_a\in[0,L]^\ell$ and $\boldsymbol{\vartheta}\in[0,L]^\ell$. We use Monte Carlo simulation to estimate the average cumulative regret $\bar{R}_T(\mathcal{D})$ over the whole parameter space. We consider M rounds of simulation. In each round of the simulation, we generate parameters $\boldsymbol{x}_a$, $\boldsymbol{y}_a$, $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$ uniformly at random from the parameter space Θ. Then we use these samples of the parameters to simulate the model, i.e., generate synthetic data. We run the algorithm on the simulated model, to evaluate its regret. Through this, we obtain one sample of the regret. Finally, we use the sample average as an estimator of the average cumulative regret over the whole parameter space. We set the number of simulation rounds to be $M=1000$.

B. Impact of the Number of Iterations N

Recall that N denotes the number of iterations for the MCMC to generate approximate samples for the posterior distribution. We study the impact of N on the regret of the GCL-PSMC. We set the dimension of the observed and latent feature as $d=5$ and $\ell=5$ respectively. Fig. 1(a) shows the regret of GCL-PSMC algorithm, where N varies

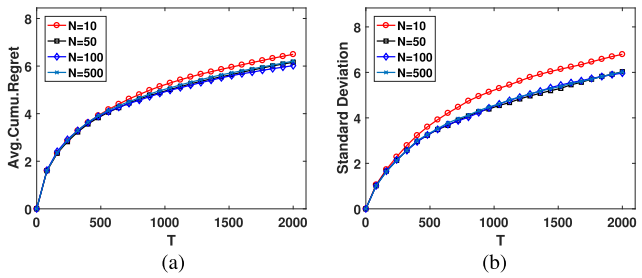


Fig. 1. Impact of the simulation rounds N on the average cumulative regret and standard deviation of regret of GCL-PSMC. (a) Average cumulative regret. (b) Standard deviation of regret.

from 10 to 500. The horizontal axis represents the time or decision rounds and the vertical axis shows the average cumulative regret over the whole parameter space defined in (6). Fig. 1(a) shows that the average cumulative regret of the GCL-PSMC algorithm increases in T and the average cumulative regret quickly gets flat. This further validates the sublinear regret upper bound derived in Theorem 4. As the number of iterations N that simulates the Markov chain for approximating the posterior distribution increases, the average cumulative regret of the GCL-PSMC varies slightly. This shows that our GCL-PSMC is highly efficient requiring a small number of iterations for simulating the Markov chain to approximate the posterior distribution. One reason for this high efficiency is that the GCL-PSMC algorithm uses the samples of $\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}$ in the last decision round as the initial point of the Markov chain in current decision round. Note that in each decision round, we only use one reward to update the posterior distribution. This means that the posterior distributions in two consecutive decision rounds do not differ a lot. Thus, the samples of $\boldsymbol{Y}, \boldsymbol{\theta}, \boldsymbol{\vartheta}$ in last decision round are not far from the stationary distribution (i.e., posterior distribution) in the current round. Fig. 1(b) shows the standard deviation of regret. One can observe that the standard deviation is comparable with the average regret. This implies that the variation of regret is not large. Based on Fig. 1(a), we set $N = 50$ for GCL-PSMC in later experiments.

C. Impact of Total Number of Dimensions

For all the algorithms described in Section V-A, we compare their average cumulative regret defined in (6). The total number of actions is fixed to $|\mathcal{A}| = 40$. We consider two selection of dimensions, i.e., the dimensions for the observed and latent feature are $d = 5$ and $\ell = 5$, respectively and $d = 10$ and $\ell = 10$, respectively, in order to reveal the impact of total dimensions on the regret. Fig. 2(a) shows the average cumulative regret of six algorithms in consideration when the dimensions for the observed and latent feature are $d = 5, \ell = 5$. One can observe that the average cumulative regret curve of GCL-PSMC lies at the bottom. In other words, GCL-PSMC has a smaller average cumulative regret than five baseline algorithms in consideration. The GCL-LIN, UCB-GLM and GLM-TSL (all of them ignore latent features) have significantly larger average cumulative regrets than GCL-PSMC. This implies that ignoring the latent feature would lead the algorithm to make poor decisions, and justifies the importance of taking latent features into consideration. The

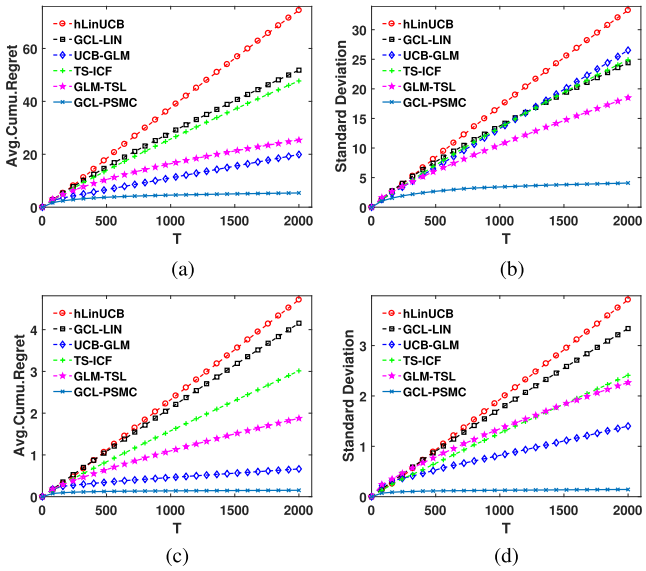


Fig. 2. Impact of number of total dimensions on the average cumulative regret and standard deviation of regret. (a) Dimension $d = \ell = 5$. (b) Dimension $d = \ell = 5$. (c) Dimension $d = \ell = 10$. (d) Dimension $d = \ell = 10$.

hLinUCB and TS-ICF (both of them ignore the nonlinearity in the reward) also has a significantly larger regret than GCL-PSMC. This implies that ignoring the nonlinearity of reward would also lead the algorithm to make poor decisions, and justifies the importance of taking the nonlinearity of reward into consideration. Fig. 2(b) shows that the standard deviation of the regret of GCL-PSMC is drastically smaller than the other five comparison baselines. This implies that GCL-PSMC is more stable, i.e., the regret of GCL-PSMC has a smaller variation than the other five comparison baselines. Fig. 2(c) and (d) shows similar observations when we increase the dimensions for the observed and latent feature to $d = 10$ and $\ell = 10$ respectively. Namely, the above statements hold when the observed and latent features have different dimensions.

D. Impact of Latent Features

1) Impact of the Range of Latent Features: We fix the dimension of observed and latent feature to $d = 5$ and $\ell = 5$, respectively. Note that the range of the latent feature and the associated preference vector lie in $\boldsymbol{y}_a \in [0, L]^\ell, \boldsymbol{\vartheta} \in [0, L]^\ell$, where $L \in \mathbb{R}_+$. We vary the range L of the latent feature and the associated preference vector from 0.5 to 4 to study its impact on the average cumulative regret. Fig. 3(a), (c), (e), and (g) shows the average cumulative regret of six algorithms in consideration. One can observe that when $L = 0.5$, i.e., the range of latent feature is small, UCB-GLM has the smallest regret and GCL-PSMC has the second smallest regret. Namely, when latent features are in a small range and thus has a small impact on the reward, UCB-GLM can outperform our GCL-PSMC. One reason is that GCL-PSMC uses MCMC to approximate the posterior distribution, and this approximation reduces exploration efficiency of the GCL-PSMC algorithm. When the range L of the latent feature is larger or equal to 1, latent features play an important role in the reward and GCL-PSMC has the

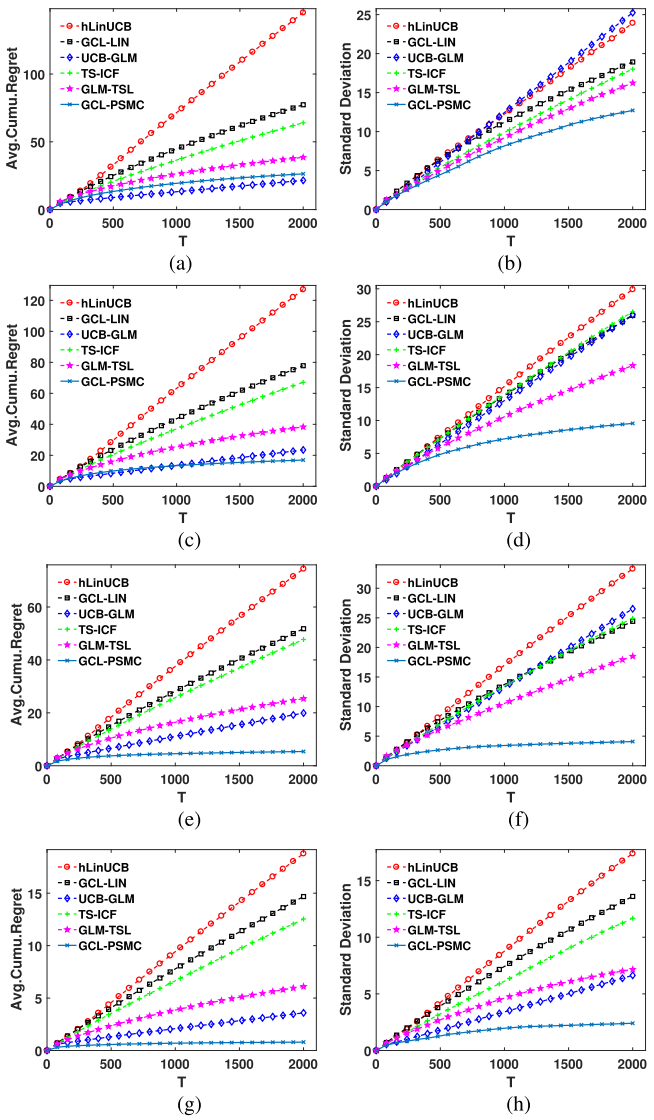


Fig. 3. Impact of the range L of latent features on the average cumulative regret and standard deviation of regret. (a) Range of latent feature $L = 0.5$. (b) Range of latent feature $L = 0.5$. (c) Range of latent feature $L = 1$. (d) Range of latent feature $L = 1$. (e) Range of latent feature $L = 2$. (f) Range of latent feature $L = 2$. (g) Range of latent feature $L = 4$. (h) Range of latent feature $L = 4$.

smallest regret. In this setting, the GCL-LIN, UCB-GLM and GLM-TSL (all of them ignore latent features), as well as hLinUCB and TS-ICF (both of them ignore the nonlinearity in the reward) can have significantly larger regret than GCL-PSMC. This result further justifies that ignoring the latent feature or nonlinearity of reward would lead the algorithm to make poor decisions. Fig. 3(b), (d), (f), and (h) shows that the standard deviation of the regret of GCL-PSMC is significantly smaller than the other five comparison baselines. This implies that GCL-PSMC is more stable, i.e., the regret of GCL-PSMC has a smaller variation than the other five comparison baselines.

2) Impact of the Dimension of the Latent Feature: We vary the dimension of the latent feature ℓ from 5 to 7 and we keep the total dimension $d + \ell = 10$. Fig. 4(a) and (c) shows the regret of six algorithms in comparison. Again, one can observe that GCL-PSMC always has the smallest regret

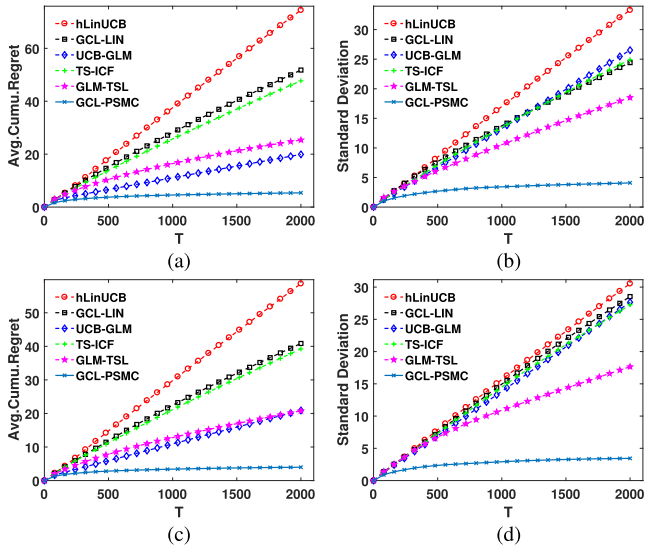


Fig. 4. Impact of the dimension ℓ of the latent feature on the average cumulative regret and standard deviation of regret. (a) Dimension of latent feature $\ell = 5$. (b) Dimension of latent feature $\ell = 5$. (c) Dimension of latent feature $\ell = 7$. (d) Dimension of latent feature $\ell = 7$.

among these six algorithms. The GCL-LIN, UCB-GLM and GLM-TSL (all of them ignore latent features), as well as hLinUCB and TS-ICF (both of them ignore the nonlinearity in the reward) have significantly larger regrets than GCL-PSMC. This result further justifies that ignoring the latent feature or nonlinearity of reward would lead the algorithm to make poor decisions and confirms the superior performance of GCL-PSMC. Fig. 4(b) and (d) shows that the standard deviation of the regret of GCL-PSMC is significantly smaller than the other five comparison baselines. This implies that GCL-PSMC is more stable, i.e., the regret of GCL-PSMC has a smaller variation than the other five comparison baselines.

VI. Applications

We apply the proposed contextual bandit framework to hotel recommendations and news article recommendations. Experiments on two public datasets further validate the superior performance of GCL-PSMC.

A. Application to Hotel Recommendation

1) Tripadvisor Dataset: We use a public Tripadvisor dataset.[1] The dataset contains ratings and reviews assigned by users to hotels. Each rating or review contains the ID of the user and the ID of the hotel. The user not only gives an overall rating to a hotel but also gives a rating to seven aspects of a hotel.

2) Experiment Setting: We use the average rating of seven aspects of a hotel as the observed feature vector of a hotel. Namely, the observed feature vector has a dimension of $d = 7$. Each user gives ratings to at most tens of hotels. We select two of them to study: 1) Jerri_Blank who assigns ratings to 16 hotels, and 2) trevostar who assigns ratings to 13 hotels. We denote Jerri_Blank and trevostar as user A and B. For each selected user, we apply our framework to recommend hotel to

[1] http://times.cs.uiuc.edu/%7Ewang296/Data/

him. The action or arm set $\mathcal{A}$ for each selected user is the set of hotels he has rated. Note that under the mapping function $g_t(V) = g(V) = 1/[1+\exp(-V)]$. Each rating r of a selected user to hotel $a \in \mathcal{A}$ is used to recover the true evaluation

$$\boldsymbol{x}_a^{\mathrm{T}}\boldsymbol{\theta} + \boldsymbol{y}_a^{\mathrm{T}}\boldsymbol{\vartheta} = g^{-1}(r/6). \tag{8}$$

We divide the rating r by 6 to make $r/6 \in (0, 1)$ as $r \in \{1, \ldots, 5\}$. Note that it is unknown to the decision maker. For each round t, we randomly select ten hotels from the set of hotels $\mathcal{A}$ that a user has rated as $\mathcal{A}_t$. Then we apply our framework to recommend a hotel from $\mathcal{A}_t$. Once a hotel is recommended, we use our model to generate feedback with the true evaluation as that derived in (8). We evaluate six algorithms as described in Section V. In the evaluation, we consider the dimension of latent features as $\ell = 3$ (or $\ell = 8$) such that the total dimension is 10 (or 15). For the unspecified parameters, we set them like that in Section V. We consider the same baselines as that in Section V.

3) Experiment Results: Fig. 5(a), (c), (e) and (g) shows the average cumulative regret of six algorithms in consideration, when we use them to recommend hotels to user A and B respectively. One can observe that the average cumulative regret curve of GCL-PSMC lies in the bottom. This means that the GCL-PSMC algorithm has the smallest the average cumulative regret among these six algorithms in comparison. Furthermore, the average cumulative regret of GCL-PSMC is around ten, while each of the other five baseline algorithms has an average cumulative regret at least one hundred, except the GLM-TSL algorithm. The average cumulative regret of the GLM-TSL is around fifty, i.e., around fives time that of GCL-PSMC. Namely, GCL-PSMC can make significantly more accurate recommendations than all these five baselines. Furthermore, the average cumulative regret curve of GCL-PSMC is flat. This implies the convergence of GCL-PSMC. Namely, it quickly can lean a user's true preference and makes accurate recommendations. In summary, the above results justify the importance of taking latent features as well as the nonlinearity of reward into considerations in the recommendation tasks. They also justify the superior performance of GCL-PSMC. Fig. 5(b) and (d) shows that the standard deviation of GCL-PSMC, GCL-LIN, and UCB-GLM are similar, and they are drastically smaller than the other three comparison baselines. Fig. 5(f) and (h) shows, that the standard deviation of GCL-PSMC and UCB-GLM are similar, and they are drastically smaller than the other four comparison baselines. This implies that GCL-PSMC is roughly the most stable one, i.e., the regret of GCL-PSMC has nearly the smallest variation among six algorithms in consideration. Taking the average regret into consideration, Fig. 5(b) and (d) also shows that the average regret of GCL-PSMC plus three times its standard deviation is smaller than the average regret of UCB-GLM, GCL-LIN, hLinUCB and TS-ICF minus three times their own standard deviation. Namely, we have pretty high confidence that the regret of GCL-PSMC is smaller than that of UCB-GLM, GCL-LIN, hLinUCB, and TS-ICF. However, for the GLM-TSL, we do not have such high confidence, as its average regret is close to GCL-PSMC and its standard deviation is not small. Similar observations can be observed in Fig. 5(f) and (h).

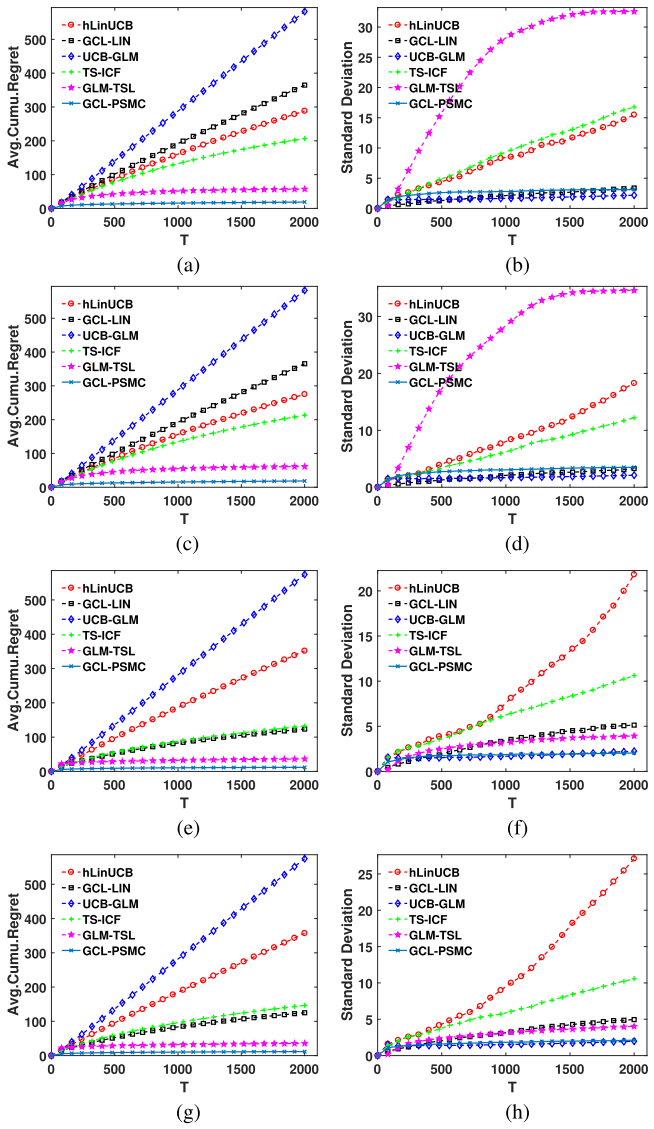


Fig. 5. Recommending hotels to users in Tripadvisor. (a) Latent dimension $\ell = 3$ (user A). (b) Latent dimension $\ell = 3$ (user A). (c) Latent dimension $\ell = 8$ (user A). (d) Latent dimension $\ell = 8$ (user A). (e) Latent dimension $\ell = 3$ (user B). (f) Latent dimension $\ell = 3$ (user B). (g) Latent dimension $\ell = 8$ (user B). (h) Latent dimension $\ell = 8$ (user B).

B. Application to News Recommendation

1) Dataset: We use a public dataset released by the Yahoo Webscope program.[2] This is a widely used benchmark dataset in testing the performance of bandit algorithms [3], [11], [22]. This dataset was collected in May 2009 from the Yahoo Today Module on Yahoo! Front Page. It contains a total number of 45 811 883 users' visits in a 10-day period. Each time when a user visits the Yahoo Today Module, the system randomly selects a news article from a pool of 20 candidate news articles to recommend. Then the user may or may not click the recommended news article. The dataset contains users' click logs in each visit. In particular, each visit of a user corresponds

[2]https://webscope.sandbox.yahoo.com/

to one item in the dataset. Each item contains a list of the IDs of 20 candidate news articles in the pool, the feature vector of each candidate news article (each feature vector is of six dimensions), the ID of the recommended news article, and the outcome indicating whether a user clicks (1) the recommended article or not (0). Note that the news article pool changes over time. Due to privacy considerations, the data item does not contain the ID information of the user.

2) Experiment Setting: As each data item does not contain the ID information of the user, we do not know which data item corresponds to which user. We thus use the data to simulate the scenario of recommending news articles to the whole user population. In other words, we treat the whole user population as a superuser. The objective of a learning algorithm is to attain a high average click rate over the whole user population, or a high click rate from the super user. Each visit of a user to the Yahoo Today Module corresponds to generating a user sample from the user population to visit the Yahoo Today Module. If a user clicks (or does not click) the recommended news article, we treat it as the superuser clicks (or does not click) the recommended news article. We use the policy evaluation developed in [22] to evaluate the click rate of an algorithm from the dataset. The idea is that given a recommendation of a news article by an algorithm (e.g., GCL-PSMC), we search the dataset from old to new to see whether there is a recommendation of the same news article. If we find one, then we use the corresponding click outcome as the reward for the algorithm. Then, we discard all the data items older than the matched data item. With the reward, the algorithm updates its parameters and then makes another round of recommendations. We repeat the same process over the remaining data items. The click rate of the algorithm is the average reward from the matched data item in the recommendation process.

We set the dimension of observed features as $d = 6$ in consistent with the data. We set the dimension of latent features as $\ell = 4$ making $d + \ell = 10$ and $\ell = 9$ making $d + \ell = 15$. We evaluate six algorithms as described in Section V. Note that under the mapping function $g_t(V) = 1/[1 + \exp(-V)]$, the outcome 0 in the dataset corresponds to $V = -\infty$ and the outcome 1 in the dataset corresponds to $V = +\infty$. Thus, for GCL-PSMC, we feed it with the mapping function $g_t(V) = 2/[1 + \exp(-V)] - 0.5$.

3) Experimental Results: Fig. 6(a) and (c) shows the click rate of six algorithms in comparison. For each algorithm in comparison, we set its parameters as that in Section V. The horizontal axes of Fig. 6(a) and (c) represent the number of recommendation rounds made by each algorithm. And the vertical axis shows the average click rate up to that time slot. One can observe that when T is less than 800, GCL-PSMC has almost the highest click rate among these six algorithms in comparison. This implies that GCL-PSMC has a fast learning speed in the early stage. When T is large than 800, GCL-PSMC has the second highest average click rate and hLinUCB has the highest average click rate but it is comparable to GCL-PSMC. One reason is that GCL-PSMC uses MCMC to approximate the posterior distribution, and this approximation reduces the long-term accuracy of the

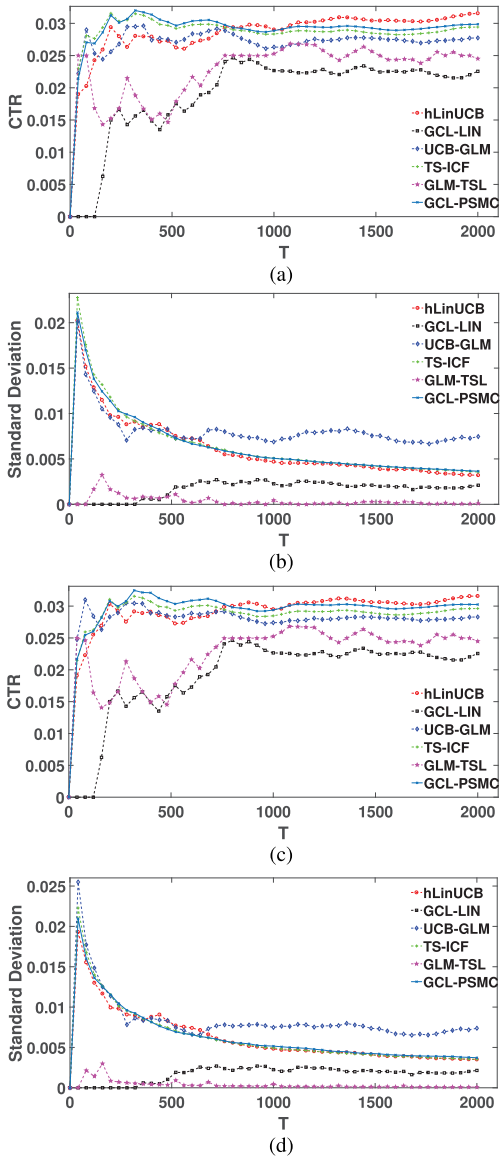


Fig. 6. Click rate evaluated from a Yahoo! dataset. (a) Latent dimension $\ell = 4$. (b) Latent dimension $\ell = 4$. (c) Latent dimension $\ell = 9$. (d) Latent dimension $\ell = 9$.

GCL-PSMC algorithm. The average click rate becomes stable, as the number of rounds T gets larger. This shows the convergence of the GCL-PSMC algorithm. The above results further validate the superior performance of our GCL-PSMC for new article recommendations. Fig. 6(b) and (d) shows the standard deviation of the click rate. One can observe that hLinUCB and GCL-PSMC have nearly the same standard deviation. Namely, these two algorithms are comparable stable. GCL-LIN and GLM-TSL have smaller standard deviation than GCL-PSMC, but their click rate is significantly smaller than GCL-PSMC. It is difficult to assure a high confidence comparison on the click rate, as the standard deviation is not small compared with the average click rate gap.

VII. Conclusion

This article presents a generalized contextual bandit framework with the latent feature. We develop a *GCL-PS* algorithm, which uses a posterior sampling algorithm to balance the

exploration and *exploitation* tradeoff for the proposed framework. We prove a sublinear Bayesian regret upper bound for *GCL-PS* and prove a lower bound for the proposed framework, revealing insights on the optimality of *GCL-PS*. To improve the computational efficiency of the *GCL-PS*, we propose a Markov Chain Monte Carlo (MCMC) algorithm to generate approximate samples, leading to the *GCL-PSMC* algorithm. We not only prove a sublinear regret upper bound for the *GCL-PSMC* algorithm, but also reveal insights on the tradeoff between computational efficiency and sequential decision accuracy. We conduct extensive experiments on synthetic data and two public datasets to show the superior performance of the *GCL-PSMC* algorithm over a variety of baselines.

REFERENCES

[1] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, 2012.

[2] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 208–214.

[3] H. Wang, Q. Wu, and H. Wang, "Learning hidden features for contextual bandits," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 1633–1642.

[4] L. Zhou and E. Brunskill, "Latent contextual bandits and their application to personalized recommendations for new users," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 3646–3653.

[5] S. Lamprier, T. Gisselbrecht, and P. Gallinari, "Contextual bandits with hidden contexts: A focused data capture from social media streams," *Data Mining Knowl. Discovery*, vol. 33, no. 6, pp. 1853–1893, Nov. 2019.

[6] S. Filippi, O. Cappé, A. Garivier, and C. Szepesvári, "Parametric bandits: The generalized linear case," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2010, pp. 586–594.

[7] L. Li, Y. Lu, and D. Zhou, "Provably optimal algorithms for generalized linear contextual bandits," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2071–2080.

[8] A. Krishnamurthy, Z. S. Wu, and V. Syrgkanis, "Semiparametric contextual bandits," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2776–2785.

[9] B. Kveton, M. Zaheer, C. Szepesvari, L. Li, M. Ghavamzadeh, and C. Boutilier, "Randomized exploration in generalized linear bandits," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2066–2076.

[10] Y. Peng *et al.*, "A practical semi-parametric contextual bandit," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 3246–3252.

[11] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, 2010, pp. 661–670.

[12] M.-H. Oh and G. Iyengar, "Thompson sampling for multinomial logit contextual bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 1–11.

[13] B. Dumitrascu, K. Feng, and B. E. Engelhardt, "PG-TS: Improved Thompson sampling for logistic contextual bandits," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4629–4638.

[14] M. Guan and H. Jiang, "Nonparametric stochastic contextual bandits," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 3119–3125.

[15] S. Li, A. Karatzoglou, and C. Gentile, "Collaborative filtering bandits," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2016, pp. 539–548.

[16] C. Zeng, Q. Wang, S. Mokhtari, and T. Li, "Online context-aware recommendation with time varying multi-armed bandit," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 2025–2034.

[17] X. Zhang, H. Xie, H. Li, and J. C. S. Lui, "Conversational contextual bandit: Algorithm and application," in *Proc. Web Conf.*, Apr. 2020, pp. 662–672.

[18] S. Li, W. Chen, S. Li, and K.-S. Leung, "Improved algorithm on online clustering of bandits," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2923–2929.

[19] C. Gentile, S. Li, P. Kar, A. Karatzoglou, G. Zappella, and E. Etrue, "On context-dependent clustering of bandits," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1253–1262.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[21] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 127–135.

[22] L. Li, W. Chu, J. Langford, T. Moon, and X. Wang, "An unbiased offline evaluation of contextual bandit algorithms with generalized linear models," in *Workshop On-line Trading Exploration Exploitation*. PMLR, 2012.

[23] D. Russo and B. Van Roy, "Learning to optimize via posterior sampling," *Math. Oper. Res.*, vol. 39, no. 4, pp. 1221–1243, Nov. 2014.

[24] H. Wang, Q. Wu, and H. Wang, "Factorization bandits for interactive recommendation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017, pp. 2695–2702.

[25] J. Kawale, H. Bui, B. Kveton, L. T. Thanh, and S. Chawla, "Efficient Thompson sampling for online? Matrix-factorization recommendation," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1297–1305.

[26] Q. Wang *et al.*, "Online interactive collaborative filtering using multi-armed bandit with dependent arms," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1569–1580, Aug. 2019.

[27] X. Zhao, W. Zhang, and J. Wang, "Interactive collaborative filtering," in *Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage. (CIKM)*, 2013, pp. 1411–1420.

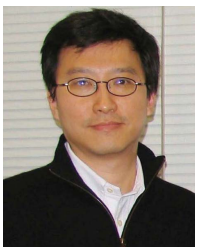
[28] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. Cham, Switzerland: Springer, 2013.

Xiongxiao Xu received the B.Eng. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2020.

He is currently a Research Assistant with the College of Computer Science, Chongqing University, advised by Prof. H. Xie. His research interests include online learning and recommender systems.

Hong Xie (Member, IEEE) received the B.Eng. degree from the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2010, and the Ph.D. degree with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. J. C. S. Lui.

He was a Post-Doctoral Research Fellow with The Chinese University of Hong Kong and National University of Singapore. He is currently a Research Professor with the College of Computer Science, Chongqing University, Chongqing, China.



John C. S. Lui (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles, CA, USA, in 1992.

He was a Chairman with the CSE Department from 2005 to 2011. He is currently the Choh-Ming Li Chair Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His current research interests include communication networks, system security (e.g., cloud security, mobile security, etc.), network economics, network sciences, large-scale distributed systems, and performance evaluation theory.

Dr. Lui is an Elected Member of the IFIP WG 7.3, a fellow of a Croucher Senior Research Fellow. He was a recipient of the various departmental teaching awards and the CUHK Vice-Chancellors Exemplary Teaching Award. He was also a co-recipient of the IFIP WG 7.3 Performance 2005 and IEEEIFIP NOMS 2006 Best Student Paper Awards. He serves in the Editorial Board for the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.