Marginal Posterior Sampling for Slate Bandits

Maria Dimakopoulou , **Nikos Vlassis** , **Tony Jebara**
Netflix
{mdimakopoulou, nvlassis, tjebara}@netflix.com

Abstract

We introduce a new Thompson sampling-based algorithm, called marginal posterior sampling, for online slate bandits, that is characterized by three key ideas. First, it postulates that the slate-level reward is a monotone function of the marginal unobserved rewards of the base actions selected in the slates's slots, but it does not attempt to estimate this function. Second, instead of maintaining a slate-level reward posterior, the algorithm maintains posterior distributions for the marginal reward of each slot's base actions and uses the samples from these marginal posteriors to select the next slate. Third, marginal posterior sampling optimizes at the slot-level rather than the slate-level, which makes the approach computationally efficient. Simulation results establish substantial advantages of marginal posterior sampling over alternative Thompson sampling-based approaches that are widely used in the domain of web services.

1 Introduction

In the traditional multi-armed bandit problem [Lai and Robbins, 1985], an agent is presented with a set of possible actions. In a sequence of repetitions, first the world generates a reward for each one of the possible actions, then the agent chooses an action among the possible ones and observes a reward. There is partial feedback, meaning that the agent observes a reward only for the action it chose, not for any other action it could have chosen. Additionally, the agent is unsure about the underlying dynamics of the world and therefore is unsure about which action is best. In this setting, there is a need to balance the exploration of arms for which there is limited knowledge in order to improve performance in the future against the exploitation of existing knowledge in order to attain better performance in the present.

Many problems in the domain of web-services, such as e-commerce and streaming, entail the selection of not only one but of multiple actions at the same time. Upon the agent's choice, a collective reward characterizing the quality of the entire selection is observed, but the rewards of the individual actions are not necessarily observed. This paper considers the slate bandit problem (also referred to as the combinatorial bandit problem [Cesa-Bianchi and Lugosi, 2012]), which is similar to the traditional multi-armed bandit problem, except that the agent selects a slate consisting of multiple slots, each one of which is occupied by an action. The number of possible slates can be combinatorially large and this is a key challenge for learning in this setting.

Slate bandits are poised to play an important role in a wide range of applications. For instance, consider the construction of the homepage of a streaming service. The streaming service sequentially selects homepages to display to users. The homepage can be viewed as a slate with slots corresponding to sections (e.g., first row, second row, etc.) and base actions corresponding to thematically coherent sets of videos (e.g. comedy, romance, etc.) to be placed in these sections. The goal is to construct a homepage that maximizes the probability of a user playing something from one of the homepage's rows. Homepage optimization can be viewed as a slate bandit problem. Similarly, the online optimization of an email campaign can also be addressed through slate bandits. In an email campaign, a decision-maker sequentially selects emails to send to users. Each email consists of several components (e.g., subject line, text, visuals) and each component has multiple options (e.g., interrogative, informative or amusing subject line, level of detail in text, whether or not to include visuals in the email). The goal of the decision-maker is to identify the treatment, i.e., the combination of options for the email components, that maximizes the probability of a user taking a desirable action (e.g., making a donation, starting a survey and so forth).

Thompson sampling [Thompson, 1933] has emerged as an effective and popular heuristic for balancing exploration and exploitation in multi-armed bandits [Russo *et al.*, 2018]. To select an action, Thompson sampling samples a model of the system from the prevailing posterior distribution and then determines which action maximizes the expected immediate reward according to the sampled model. We propose a new Thompson sampling-based algorithm, called *marginal posterior sampling*, for online slate bandits, that is characterized by three key ideas. First, it postulates that the slate-level reward is a non-decreasing function of the marginal unobserved rewards of the base actions occupying the slate's slots. However, it does not attempt to estimate this function. Second, instead of maintaining a slate-level reward posterior, the algorithm maintains posterior distributions for the marginal rewards of each slot's base actions and uses the samples from these marginal posteriors to select the next slate. Third, marginal posterior sampling optimizes at the slot-level rather than the slate-level, which makes it computationally very efficient.

2 Related Work

Multiple prior works have considered the slate bandit problem with semi-bandit feedback – where the rewards of all slate's actions are revealed to the agent – and have developed online slate bandit algorithms [Lagrée *et al.*, 2016; Li *et al.*, 2016; Wang *et al.*, 2017] or off-policy evaluation estimators for slate policies [Li *et al.*, 2018]. In several works that assume semi-bandit feedback, the optimization is performed at the slate level, either iterating over the entire slate space or assuming that the agent has access to an ϵ-approximation oracle ($\epsilon < 1$) that, given value estimates for each slot-action pair, outputs the slate that has estimated value at least ϵ times the estimated value of the estimated optimal slate [Chen *et al.*, 2013; Qin *et al.*, 2014; Li *et al.*, 2016]. The work of [Agrawal *et al.*, 2017] deals with a similar problem in which slates are assortments, the user can select a single item from the assortment and the reward of each item in the assortment is revealed. Similar to our approach, in [Agrawal *et al.*, 2017] separate marginal distributions are maintained for the base actions, and a technique is described for updating the marginals by repeatedly showing the same slate until a negative response is received. Our approach can be regarded as a constrained assortment problem where each slot can only accept base actions from the action space of that slot, and the learner receives feedback on *sets* of base actions instead of a single action. An interesting question is whether the same technique of [Agrawal *et al.*, 2017] can be employed for more general combinatorial bandit problems, like the one we study here.

The most important difference between these works and ours is the fact that we only assume observation of the slate-level reward and the rewards of slate's base actions do not need to be observed. Moreover, we avoid optimization over the entire slate space – which can be potentially huge – and the need of an approximation oracle by optimizing at the slot-level rather than the slate-level.

Our assumption that the slate-level reward can be represented via a non-decreasing link function of the marginal unobserved rewards of the slate's base actions plays the same role as the additivity assumption of [Kale *et al.*, 2010] in the online semi-bandit problem and of [Swaminathan *et al.*, 2017] in the offline evaluation of policies with slate-level feedback. However, the non-decreasing link function assumption we make is significantly weaker than the additive link function assumption common in the literature.

The slate bandit problem can be viewed as a bandit with arms with covariates, where the covariates of an arm (slate) is the concatenation of the one-hot encodings of each slot's base action. Two popular methods of addressing slate bandit problems in practice are linear bandits [Li *et al.*, 2010; Agrawal and Goyal, 2013] and generalized linear bandits (particularly logistic) [Chapelle and Li, 2011; Li *et al.*, 2017] which postulate that the reward of an arm can be modeled as a linear or a generalized linear function of its features with unknown parameters. These algorithms use regularized linear or logistic regression to form an upper confidence bound or a posterior (exact or approximate) on the unknown parameters. Subsequently, the upper confidence bound or the posterior is used to balance exploration vs. exploitation when deciding the next arm. Although these methods are wide-spread in solving slate bandit problems in web-services, ranging from search to news recommendations to advertising, there are certain drawbacks. First, they attempt to maintain a posterior or an upper confidence bound at the slate level by making a functional form assumptions that in practice may be misspecified and may impede learning performance [Dimakopoulou *et al.*, 2019]. Second, the choice of the slate requires optimization over the entire slate space, which can be potentially intractable.

Marginal posterior sampling is designed to address both these issues. In a range of experiments we demonstrate that marginal posterior sampling has significantly better performance in terms of cumulative reward compared to generalized linear bandits, improving up to 30%. At the same time, marginal posterior sampling can make a slate decision up to 70 times faster than generalized linear bandits.

3 Problem Formulation

We formulate the problem of an online slate bandit as follows. A slate $\mathbf{s} = (s_1, \ldots, s_\ell)$ consists of base actions s_j, where a position $j \in \{1, \ldots, \ell\}$ is called a slot. There are ℓ slots in the slate. The action of slot j comes from an action space $\mathcal{A}_j$ with cardinality $|\mathcal{A}_j| = m_j$ and we define $m = \max_j m_j$.

A slate $\mathbf{s}$ can be represented by a binary feature vector $\mathbf{x_s} \in \{0,1\}^{\ell m}$, whose components are indexed by pairs (j, a) of slots and possible base actions in them. The entry at position (j, a) is equal to 1 if the slate $\mathbf{s}$ has action a at slot j, i.e., if $s_j = a$. The space of possible slates is $\mathcal{S} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_\ell$.

Given a slate $\mathbf{s}$, a reward $r \in \{0, 1\}$ is drawn from a distribution $\mathcal{D}(r|\mathbf{s})$. The expected reward for a given slate is referred to as the slate value and is denoted as: $V(\mathbf{s}) = \mathbb{E}_{r \sim \mathcal{D}(r|\mathbf{s})}[r]$

In the example of a streaming service's homepage optimization, there are ℓ positions in the homepage which correspond to the slots of the slate. The action space of each slot is the same m sets of videos ($m > \ell$), which correspond to the base actions of the slate. If no repetitions are allowed in the positions of the homepage, the number of valid slates is $|\mathcal{S}| = \frac{m!}{(m-\ell)!}$ and this problem is equivalent to ranking. The reward of slate $\mathbf{s}$ is whether the user played a video from one of the ℓ positions when presented with homepage $\mathbf{s}$.

In the example of optimizing an email campaign, there are ℓ components in an email (e.g., subject line, text, visuals) which correspond to the slots of the slate. The action space of each slot is different (e.g., interrogative, informative or amusing subject line, level of detail in the text, whether or not to include visuals). The set of all possible emails is the cartesian product $\mathcal{S} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_\ell$ and $|\mathcal{S}| = \prod_{j=1}^{\ell} \mathcal{A}_j$. The reward of slate $\mathbf{s}$ is whether the recipient clicked on email $\mathbf{s}$.

Algorithm 1 Slate Bandit as K-Armed Bernoulli Bandit

```
Require: Initial $\alpha(\mathbf{s})$ and $\beta(\mathbf{s})$ for all $\mathbf{s} \in \mathcal{S}$ (default value: 1)
for $t = 1, \dots, T$ do
    for each slate $\mathbf{s} \in \mathcal{S}$ do
        Sample $\hat{\theta}(\mathbf{s}) \sim$ Beta $(\alpha(\mathbf{s}), \beta(\mathbf{s}))$
    end for
    Select slate $\mathbf{s}_t = \mathrm{argmax}_{\mathbf{s}} \hat{\theta}(\mathbf{s})$
    Observe reward $r_t \sim \mathcal{D}(r|\mathbf{s}_t)$
    if $r_t = 1$ then
        $\alpha(\mathbf{s}_t) = \alpha(\mathbf{s}_t) + 1$
    else
        $\beta(\mathbf{s}_t) = \beta(\mathbf{s}_t) + 1$
    end if
end for
```

The decision-maker sequentially chooses slates $(\mathbf{s}_t)_{t \in \mathbb{N}}$ from the slate space $\mathcal{S}$ and observes the corresponding reward $(r_t)_{t \in \mathbb{N}}$. The rewards observed by the agent in different time periods are independent, conditioned on the chosen slates. Denote as $\mathbf{s}^* \in \mathrm{argmax}_{\mathbf{s} \in \mathcal{S}} V(\mathbf{s})$ the optimal slate. The goal of the agent is to minimize the expected regret over a horizon T, $\mathbb{E}[\text{Regret}(T)] = \sum_{t=1}^{T} [V(\mathbf{s}^*) - V(\mathbf{s}_t)]$ which measures the cumulative difference between the reward earned by an algorithm that always chooses the optimal slate and the actual accumulated reward up to time T.

4 Algorithms

First, we discuss the straight-forward adaptation of the K-armed Bernoulli bandit – which is one of the simplest multi-armed bandit algorithms – and of the generalized linear bandit with covariates – which is widely used in web services applications – to the slate bandit problem. Subsequently, we present our proposed approach, marginal posterior sampling for slate bandits. We focus on slate bandit algorithms for slates with binary rewards, but the proposed approach can be readily extended to real number rewards, e.g., by using Gaussian rather than Beta posteriors.

4.1 Baselines

K-Armed Bernoulli Bandit

One straightforward approach is to model the slate bandit as a K-armed Bernoulli bandit with independent arms, where each slate corresponds to an arm, i.e. $K = |\mathcal{S}|$. In this formulation, the reward of arm $\mathbf{s}$ follows a Bernoulli distribution with mean $\theta(\mathbf{s})$. It is standard to model the mean reward of arm $\mathbf{s}$ using a Beta distribution with parameters $\alpha(\mathbf{s})$ and $\beta(\mathbf{s})$, since it is the conjugate distribution of the binomial distribution. At every time t, the agent draws a mean reward $\hat{\theta}(\mathbf{s}) \sim$ Beta $(\alpha(\mathbf{s}), \beta(\mathbf{s}))$ for each slate $\mathbf{s} \in \mathcal{S}$ and plays slate $\mathbf{s}_t = \mathrm{argmax}_{\mathbf{s}} \hat{\theta}(\mathbf{s})$. Algorithm 1 presents the approach.

One issue with this approach is that it considers the slates as independent, when in fact they are not. For instance, emails with the same subject line or text are likely to have similarities in their success probabilities. A reward observation for slate $\mathbf{s}'$ could be used to augment the agent's knowledge of a slate $\mathbf{s} \neq \mathbf{s}'$, when there is overlap in the base actions of some of slates' $\mathbf{s}$ and $\mathbf{s}'$ slots, i.e., if there is a slot subset $\mathcal{J} \subseteq \{1, \dots, \ell\}$ such that $s_j = s'_j$ for each $j \in \mathcal{J}$. Casting the slate bandit as a K-armed Bernoulli bandit with independent arms misses the opportunity to learn across slates, because the beliefs of a slate's $\mathbf{s}$ reward are updated only based on observations of slate $\mathbf{s}$ and not based on observations of different but overlapping slates. Hence, learning is decelerated, particularly when there is a large number of slates.

Algorithm 2 Slate Bandit as Generalized Linear Bandit

```
Require: Parameters of weight prior $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$
Draw weight sample $\hat{\mathbf{w}} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$
for $t = 1, \dots, T$ do
    for each slate $\mathbf{s} \in \mathcal{S}$ do
        Compute $\hat{\theta}(\mathbf{s}) = \frac{1}{1+\exp(-\hat{\mathbf{w}}^\top \mathbf{x}_\mathbf{s})}$
    end for
    Select slate $\mathbf{s}_t = \mathrm{argmax}_{\mathbf{s}} \hat{\theta}(\mathbf{s})$
    Observe reward $r_t \sim \mathcal{D}(r|\mathbf{s}_t)$
    Update weight posterior parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$
    Draw a new weight sample $\hat{\mathbf{w}} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$
end for
```

Another issue is that when deciding which slate to send, this algorithm sweeps the entire slate space, which can be computationally inefficient.

Slate Bandit as Generalized Linear Bandit

Linear bandits [Li *et al.*, 2010; Agrawal and Goyal, 2013] and generalized linear bandits (particularly logistic) [Chapelle and Li, 2011; Li *et al.*, 2017] are widely used in web services from news recommendation to advertising to search. Generalized linear bandits (logistic regression in particular) have demonstrated stronger performance than linear bandits in many applications where rewards are binary.

In this section, we model the slate bandit problem as a generalized linear bandit, as in [Chapelle and Li, 2011]. As mentioned in Section 3, a slate $\mathbf{s}$ can be represented by a binary feature vector $\mathbf{x}_\mathbf{s} \in \{0, 1\}^{\ell m}$ whose entry at position (j, a) is equal to 1 if $s_j = a$.

The agent models the expected reward of slate $\mathbf{s}_t$ as a logistic function of $\mathbf{x}_{\mathbf{s}_t}$ with unknown weights $\mathbf{w} \in \mathbb{R}^{\ell m}$, $\theta(\mathbf{s}_t) = \mathbb{P}(r_t = 1|\mathbf{x}_{\mathbf{s}_t}) = \sigma(\mathbf{w}^\top \mathbf{x}_{\mathbf{s}_t})$ where $\sigma(x) \equiv \frac{1}{1+\exp(-x)}$ is the sigmoid function.

The posterior distribution on the weights is approximated by a multivariate Gaussian distribution updated via the Laplace approximation. Specifically, the agent starts with a multivariate Gaussian prior over $\mathbf{w}$ with mean $\boldsymbol{\mu}_0 = \mathbf{0} \in \mathbb{R}^{\ell m}$ and covariance matrix $\boldsymbol{\Sigma}_0 = \lambda \cdot \mathbb{I}_{\ell m}$, where $\mathbb{I}_{\ell m}$ is the $\ell m \times \ell m$ identity matrix and λ is a regularization parameter. The log-posterior of $\mathbf{w}$ at time t is

$$\log(\mathbb{P}(\mathbf{w}|\mathbf{x}_{\mathbf{s}_t}, r_t)) \propto -\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_{t-1})^\top \boldsymbol{\Sigma}_{t-1}^{-1}(\mathbf{w} - \boldsymbol{\mu}_{t-1}) + \\ + r_t \log\left(\sigma(\mathbf{w}^\top \mathbf{x}_{\mathbf{s}_t})\right) + (1 - r_t) \log\left(1 - \sigma(\mathbf{w}^\top \mathbf{x}_{\mathbf{s}_t})\right)$$

The posterior mean of $\mathbf{w}$ is the maximum a posteriori estimate $\boldsymbol{\mu}_t = \mathbf{w}_{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{w}} \log(\mathbb{P}(\mathbf{w}|\mathbf{X}, \mathbf{r}))$ and the posterior covariance matrix of $\mathbf{w}$ is $\boldsymbol{\Sigma}_t^{-1} = \boldsymbol{\Sigma}_{t-1}^{-1} + \sigma(\mathbf{w}_{\mathrm{MAP}}^\top \mathbf{x}_{\mathbf{s}_t})(1 - \sigma(\mathbf{w}_{\mathrm{MAP}}^\top \mathbf{x}_{\mathbf{s}_t}))\mathbf{x}_{\mathbf{s}_t} \mathbf{x}_{\mathbf{s}_t}^\top$. To choose the next slate, the agent draws a weight sample $\hat{\mathbf{w}} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ and forms an estimate of the expected reward $\hat{\theta}(\mathbf{s}) = \sigma(\hat{\mathbf{w}}^\top \mathbf{x}_\mathbf{s})$ of each slate $\mathbf{s} \in \mathcal{S}$ based on this weight sample. Then, the agent plays slate $\mathbf{s}_t = \operatorname{argmax}_\mathbf{s} \hat{\theta}(\mathbf{s})$. Algorithm 2 outlines the approach.

4.2 Marginal Posterior Sampling for Slate Bandits

We propose a new Thompson-sampling based algorithm for slate bandits, called *marginal posterior sampling*. There are three key ideas behind our approach:

1. The slate-level reward is assumed to be a non-decreasing function of the marginal unobserved rewards of the base actions selected in the slates's slots, but this function does not need to be estimated.
2. Instead of maintaining a slate-level reward posterior, marginal posterior sampling maintains posterior distributions for the marginal reward of each slot's base actions and uses the samples from these marginal posteriors to select the next slate.
3. The optimization for selecting a slate in every time period happens at the slot-level rather than the slate-level, which makes the approach computationally efficient.

Marginal posterior sampling assumes that the slate-level expected reward is a non-decreasing function of unobserved action-level expected rewards at each one of the slate's slots.

Assumption 1 (Slate value decomposition)**.** *There exists (unknown) function f and (unknown) reward vector $\boldsymbol{v} \in [0,1]^{\ell m}$ for the unobserved action-level rewards at each one of slate's $\boldsymbol{s}$ slots, such that the slate value satisfies*

$$V(\boldsymbol{s}) = f\left(v(1,s_1),\dots,v(\ell,s_\ell)\right)$$

Furthermore, f is non-decreasing in $\boldsymbol{v}$, i.e., for any $\boldsymbol{v},\boldsymbol{v}' \in [0,1]^{\ell m}$ such that $v(j,a) \leq v'(j,a)$ for all (j,a), $f\left(v(1,s_1),\dots,v(\ell,s_\ell)\right) \leq f\left(v'(1,s_1),\dots,v'(\ell,s_\ell)\right)$.

For example, consider a function f which posits that the slate-level success probability is a sigmoid function of the sum of its slot-action success probabilities, $f\left(v(1,s_1),\dots,v(\ell,s_\ell)\right) = \sigma\left(v(1,s_1)+\cdots+v(\ell,s_\ell)\right)$. This function f trivially satisfies the above assumption. Our assumption is much weaker than the additivity assumption, common in the literature [Kale *et al.*, 2010; Swaminathan *et al.*, 2017], which posits that the slate-level expected reward is the sum of unobserved action-level expected rewards at each one of the slate's slots.

Instead of maintaining a slate-level reward posterior as in the K-armed Bernoulli bandit and the generalized linear bandit approach of Section 4.1, marginal posterior sampling models the unobserved marginal reward of a slot's base actions using a Beta distribution. Specifically, for the unobserved reward of each slot-base action pair (j,a), the algorithm maintains a Beta posterior with parameters $\alpha(j,a)$ and $\beta(j,a)$ initialized at 1.

When slate $\mathbf{s}$ is played and reward r is observed, marginal posterior sampling relies on the non-decreasing link function assumption (Assumption 1) to impute the unobserved reward of each slot's action as $\tilde{r}(j,s_j) = r$. With this imputation, the Beta distributions of the slate's slot-action pairs are updated as follows: $\alpha(j,s_j) = \alpha(j,s_j) + \tilde{r}(j,s_j)$ and $\beta(j,s_j) = \beta(j,s_j) + (1-\tilde{r}(j,s_j))$ for all $j = 1,\dots,\ell$. Therefore, at any given time t the marginal posterior of the unobserved reward of slot-action pair (j,a) is a Beta distribution with parameters $\alpha(j,a) = 1 + \sum_{\tau=1}^{t} \mathbf{1}\{s_{\tau j} = a\} r_\tau$, $\beta(j,a) = 1 + \sum_{\tau=1}^{t} \mathbf{1}\{s_{\tau j} = a\}(1-r_\tau)$.

Algorithm 3 Marginal Posterior Sampling

```
Require: Initial $\alpha(j,a)$ and $\beta(j,a)$ for all $(j,a) \in \{1,\dots,\ell\} \times \{1,\dots,m\}$ (default value: 1)
for $t = 1,\dots,T$ do
    for each slot $j \in \{1,\dots,\ell\}$ do
        for each base action $a \in \mathcal{A}_j$ do
            Draw sample $\hat{v}(j,a) \sim \mathrm{Beta}(\alpha(j,a),\beta(j,a))$
        end for
    end for
    Select $\mathbf{s}_t = \left(\operatorname*{argmax}_{a\in\mathcal{A}_1} \hat{v}(1,a),\dots,\operatorname*{argmax}_{a\in\mathcal{A}_\ell} \hat{v}(\ell,a)\right)$
    Observe reward $r_t \sim \mathcal{D}(r|\mathbf{s}_t)$
    for each slot $j \in \{1,\dots,\ell\}$ do
        Impute slot-action reward $\tilde{r}(j,s_{tj}) = r_t$
        if $\tilde{r}(j,s_{tj}) = 1$ then
            $\alpha(j,s_{tj}) = \alpha(j,s_{tj}) + 1$
        else
            $\beta(j,s_{tj}) = \beta(j,s_{tj}) + 1$
        end if
    end for
end for
```

At every time t, in order to make its next decision, the agent samples from the marginal posteriors of each slot's base-actions, $\hat{v}(j,a) \sim \mathrm{Beta}(\alpha(j,a),\beta(j,a))$ and uses these samples to construct a slate. By Assumption 1, since the link function mapping the unobserved slot-action-level rewards to the slate-level reward is non-decreasing, maximization of the slate is reduced to the maximization of each one of the slate's slots. Hence, the agent selects slate $\mathbf{s}_t = (s_{t1},\dots,s_{t\ell})$ where $s_{tj} = \operatorname{argmax}_{a\in\mathcal{A}_j} \hat{v}(j,a)$. Marginal posterior sampling is outlined in Algorithm 3.

Discussion

A key feature of marginal posterior sampling is the imputation of the latent base actions' rewards with the observed slate reward r and the one-step updates of the base actions' Beta distributions, which is justified by Assumption 1. Hence, marginal posterior sampling can perform Thompson sampling efficiently, even in the face of complex models for which the posterior updating is intractable.

At the same time, marginal posterior sampling does not need to make any parametric model assumption or estimate any model, as generalized linear bandits do. When the link function is given by a generalized linear model, i.e., $f\left(v(1,s_1),\dots,v(\ell,s_\ell)\right) = \sigma\left(v(1,s_1)+\cdots+v(\ell,s_\ell)\right)$, there is a direct analogy of marginal posterior sampling to the generalized linear bandits described in Section 4.1. Recall that in the latter, the slate-level response is modeled as $V(\mathbf{s}) = \sigma(\mathbf{w}^\top \mathbf{x}_\mathbf{s})$ where $\mathbf{x}_\mathbf{s}$ is the slate indicator vector. If we restrict the weights $\mathbf{w}$ to lie in $[0,1]$ and use a Beta prior (instead of Gaussian) for those, then the above is clearly identical to assuming a link function $f(v(1,s_1),\dots,v(\ell,s_\ell)) =$

$\sigma(v(1,s_1)+\cdots+v(\ell,s_\ell))$, where $v(j,a)=w(j,a)$. In both cases, that is, when using a Beta prior or a Gaussian prior on $v(j,a)$, Bayes rule is not closed-form and some approximation is in order. In the case of generalized linear bandits with a Gaussian prior the update of the marginals is approximate (typically given via a Laplace approximation). The latter introduces extraneous variance which slows down the learning process and incurs additional regret, as we show in the experiments of Section 5.

Moreover, unlike the approaches of Section 4.1, marginal posterior sampling scales well in the presence of slots with a large number of base actions or of a large number of slots or both. If there are ℓ slots with m base actions each, marginal posterior sampling has complexity $O(\ell m)$ instead of $O(m^\ell)$.

For both training and optimization, marginal posterior sampling leverages the slate value decomposition assumption. Intuitively, this assumption states that every slot contributes more or less positively to the value of the slate. Therefore, learning the best action for each slot does not harm the slate value overall. This is a reasonable assumption for most web-service applications, where the slots' action spaces are often curated to be of high quality ahead of time. Moreover, slate diversity, which is a desirable property in web-services, may be formulated as a non-decreasing, submodular function using an entropy regularizer [Qin *et al.*, 2014].

Despite its applicability, there are scenarios where this assumption does not hold, for instance, when there are negative interactions among slots. However, we argue that marginal posterior sampling imposes far weaker limitations than many slate bandit works in the literature, which assume an additive link-function [Kale *et al.*, 2010; Swaminathan *et al.*, 2017] or a parametric form that may be misspecified. Additionally, leveraging slate value decomposition, marginal posterior sampling achieves improved performance in terms of regret and decreases computational time, as shown in Section 5.

5 Simulations

We now compare marginal posterior sampling with the baselines. First, we consider a link function f mapping the unobserved slot-action-level expected rewards to the slate-level expected reward that is additive, $V(\mathbf{s})=\sum_{j=1}^{\ell} v(j,s_j)$ [1], similar to [Gyorgy, Linder, Lugosi and Ottucsak, 2007], [Kale, Reyzin, Schapire, 2010], [Swaminathan et al., 2016]. Subsequently, we consider the following link function f

$$V(\mathbf{s})=\frac{1}{1+\exp\left(-2\left(\sum_{j=1}^{\ell}\left(v(j,s_j)+\sum_{j'=1}^{\ell}c_{jj'}v(j,s_j)v(j',s_{j'})\right)-1\right)\right)}$$

where $c_{jj'}\sim\text{Uniform}([10,20])$ for all $j,j'\in\{1,\ldots,\ell\}$, which maps the unobserved slot-action-level expected rewards to the slate-level expected reward that is sigmoid and also includes random positive interactions between all slots.

In both cases, the slot-action marginal expected rewards are drawn uniformly from the interval $[0.05,0.15]$, i.e., for all $(j,a)\in\{1,\ldots,\ell\}\times\{1,\ldots,m\}$, $v(j,a)\sim\text{Uniform}([0.05,0.15])$. The size of the slate space is $|\mathcal{S}|=m^\ell$. The optimal slate is $\mathbf{s}^*=\text{argmax}_{\mathbf{s}\in|\mathcal{S}|}V(\mathbf{s})$. K-armed Bernoulli bandit, generalized linear bandit and marginal posterior sampling sequentially select slates in $\mathcal{S}$ over a horizon of $T=50000$ time periods.

For the additive link function case, Figure 1 shows the cumulative regret of each algorithm for slates with $\ell=2$ slots and $m=2,3,4,5$ base actions per slot. The results are averaged over 1000 simulations and 95% confidence intervals are shown. Table 1 shows the percentage improvement in cumulative regret of marginal posterior sampling over the generalized linear bandit, $\frac{\text{Regret}^{GLM}(T)-\text{Regret}^{MPS}(T)}{\text{Regret}^{GLM}(T)}100\%$. The results are presented for slates with $\ell=2,3,4$ slots and $m=2,3,4,5$ base actions per slot and are averaged over 1000 simulations.

	$m=2$	$m=3$	$m=4$	$m=5$
$\ell=2$	22.89%	30.18%	27.86%	28.29%
$\ell=3$	9.74%	15.44%	16.15%	21.35%
$\ell=4$	6.32%	8.13%	12.97%	14.84%

Table 1: Cumulative regret percentage improvement of marginal posterior sampling over generalized linear bandit when the link function is additive.

For the case of the sigmoid link function with random positive interactions, Figure 2 shows the cumulative regret of each algorithm for slates with $\ell=2$ slots and $m=2,3,4,5$ base actions per slot and Table 1 shows the percentage improvement in cumulative regret of marginal posterior sampling over the generalized linear bandit. Again the results are averaged over 1000 simulations.

	$m=2$	$m=3$	$m=4$	$m=5$
$\ell=2$	9.52%	21.63%	23.39%	23.52%
$\ell=3$	10.12%	14.66%	22.47%	22.82%
$\ell=4$	6.88%	9.97%	15.46%	16.43%

Table 2: Cumulative regret percentage improvement of marginal posterior sampling over generalized linear bandit when the link function is sigmoid with random positive interactions between slots.

We see that K-armed Bernoulli bandit performs the worse, as the success probability of each slate is learnt independently without benefiting from observations of overlapping slates. Marginal posterior sampling consistently performs the best and has lower regret than the generalized linear bandit for both a simple additive and a complex sigmoid link function with interactions and for a slates with varying number of slots and varying number of base actions per slot. In most cases, the improvement is more than 15% and goes up to 30%.

Finally, we compare the algorithms in terms of time required for choosing the next slate in the case of sigmoid link function with interactions. This time includes sampling from posteriors, optimizing based on these samples, and updating the posteriors. Marginal posterior sampling holds simple marginal posteriors at the slot-action level and avoids (approximate) Bayesian updating over the slate posterior, as generalized linear bandit does, which often becomes in-

[1] When responses are binary, additivity can lead to the expected reward of the slate being greater than 1, in which case it is capped. In our experiments, this does not occur.

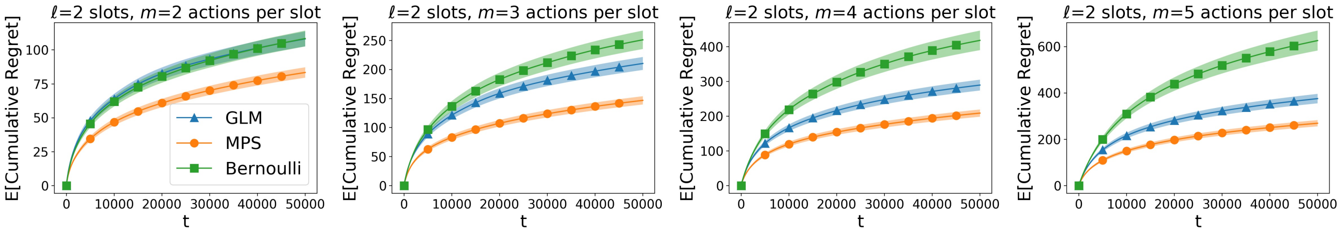

Figure 1: Cumulative regret of K-armed Bernoulli bandit, generalized linear bandit and marginal posterior sampling for the slate bandit problem with $\ell = 2$ slots, $m = 2, 3, 4, 5$ base actions per slot and an additive link function.

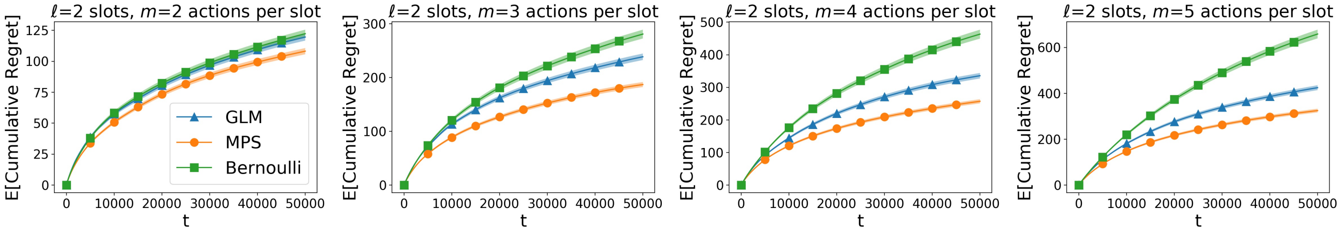

Figure 2: Cumulative regret of K-armed Bernoulli bandit, generalized linear bandit and marginal posterior sampling for the slate bandit problem with $\ell = 2$ slots, $m = 2, 3, 4, 5$ base actions per slot and a sigmoid link function with random positive interactions between slots.

$\ell = 4$	Bernoulli	GLM	MPS
$m = 2$	0.10 ms	2.42 ms	0.19 ms
$m = 3$	0.32 ms	4.82 ms	0.19 ms
$m = 4$	0.92 ms	8.73 ms	0.19 ms
$m = 5$	2.14 ms	14.25 ms	0.20 ms

Table 3: Duration of a single slate decision for K-armed Bernoulli bandit, generalized linear bandit (GLM) and marginal posterior sampling (MPS) when the link function is sigmoid with random positive interactions between slots (results averaged over 32 simulations).

tractable due to non-conjugacy. Moreover, the optimization for marginal posterior sampling is done at the slot level and is $O(m\ell)$ rather than the slate level, which is $O(m^\ell)$. As shown in Table 3, the performance of marginal posterior sampling scales gracefully for large slates and can be up to 10 times faster than K-armed Bernoulli bandit and 70 times faster than generalized linear bandit.

6 Extensions

Marginal posterior sampling can be extended to incorporate contextual information. Contextual marginal posterior sampling leverages the non-decreasing link function assumption to model the reward per slot as a function of the contextual information. If the user's contextual information $\mathbf{u}$ is d-dimensional, each slot j maintains for each one of its base actions $a \in \mathcal{A}_j$ an independent model $g_{j,a}(\mathbf{u}; \boldsymbol{\zeta}(j,a))$ parametrized by d-dimensional unknown parameters $\boldsymbol{\zeta}(j,a) \in \mathbb{R}^d$ (e.g. linear $g_{j,a}(\mathbf{u}) = \boldsymbol{\zeta}(j,a)^T\mathbf{u}$ or logistic $g_{j,a}(\mathbf{u}) = \frac{1}{1+\exp(-\boldsymbol{\zeta}(j,a)^T\mathbf{u})}$). The models $g_{j,a}$ and $g_{j,a'}$ can be trained independently, because when slot j has action a it does not have action a'. When at time t, the contextual information is $\mathbf{u}_t$, the chosen slot $\mathbf{s}_t$ has action a at slot j and the reward is r_t, the algorithm uses the non-decreasing link function assumption to impute the unobserved marginal reward as $\tilde{r}(j,a) = r_t$ and then uses the imputed observation $(\mathbf{u}_t, \tilde{r}(j,a))$ to update the posterior on $\boldsymbol{\zeta}(j,a)$ [Li *et al.*, 2010]. For selecting the slate for user $\mathbf{u}$, contextual marginal posterior sampling samples $\hat{\boldsymbol{\zeta}}(j,a)$ from the marginal posterior of each slot-action pairs' model parameters and uses it to constuct an estimate $\hat{v}(j,a) = g_{j,a}(\mathbf{u}; \hat{\boldsymbol{\zeta}}(j,a))$. Subsequently, it does slot-level optimization and selects slate $\mathbf{s} = (s_1, \ldots, s_\ell)$ where $s_j = \operatorname{argmax}_{a \in \mathcal{A}_j} \hat{v}(j,a)$.

7 Conclusion

We have presented a novel Thompson sampling approach for online slate bandits, called marginal posterior sampling. By postulating a non-decreasing link function that maps the marginal unobserved rewards at the slot-action level to the slate-level reward, marginal posterior sampling is able to maintain slot-action level posteriors and uses these marginal posteriors rather than a slate level posterior to balance exploration versus exploitation. Unlike generalized linear bandit, marginal posterior sampling does not make any functional form assumptions or estimates any function and solely relies on monotonicity, which is a much weaker assumption than the additivity assumption used in the literature. Additionally, marginal posterior sampling optimizes at the slot-level rather than the slate-level, which makes the approach computationally very efficient. We have shown the strong advantage of marginal posterior sampling compared to the generalized linear bandit – a wide-spread choice in slate problems that appear in recommendations. For slates that have up to 4 slots and up to 5 actions per slot, marginal posterior sampling can achieve up to 30% improvement in the quality of decisions and can be up to 70 times faster in its decision making.

References

[Agrawal and Goyal, 2013] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135, 2013.

[Agrawal *et al.*, 2017] Shipra Agrawal, Vashist Avadhanula, Vineet Goyal, and Assaf Zeevi. Mnl-bandit: A dynamic learning approach to assortment selection. *arXiv preprint arXiv:1706.03880*, 2017.

[Cesa-Bianchi and Lugosi, 2012] Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

[Chapelle and Li, 2011] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[Chen *et al.*, 2013] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159, 2013.

[Dimakopoulou *et al.*, 2019] Maria Dimakopoulou, Zhengyuan Zhou, Susan Athey, and Guido Imbens. Balanced linear contextual bandits. *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[Kale *et al.*, 2010] Satyen Kale, Lev Reyzin, and Robert E Schapire. Non-stochastic bandit slate problems. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2010.

[Lagrée *et al.*, 2016] Paul Lagrée, Claire Vernade, and Olivier Cappé. Multiple-play bandits in the position-based model. In *Advances in Neural Information Processing Systems*, pages 1597–1605, 2016.

[Lai and Robbins, 1985] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[Li *et al.*, 2010] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[Li *et al.*, 2016] Shuai Li, Baoxiang Wang, Shengyu Zhang, and Wei Chen. Contextual combinatorial cascading bandits. In *ICML*, volume 16, pages 1245–1253, 2016.

[Li *et al.*, 2017] Lihong Li, Yu Lu, and Dengyong Zhou. Provably optimal algorithms for generalized linear contextual bandits. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2071–2080. JMLR. org, 2017.

[Li *et al.*, 2018] Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, S Muthukrishnan, Vishwa Vinay, and Zheng Wen. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1685–1694. ACM, 2018.

[Qin *et al.*, 2014] Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 461–469. SIAM, 2014.

[Russo *et al.*, 2018] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[Swaminathan *et al.*, 2017] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*, pages 3632–3642, 2017.

[Thompson, 1933] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[Wang *et al.*, 2017] Yingfei Wang, Hua Ouyang, Chu Wang, Jianhui Chen, Tsvetan Asamov, and Yi Chang. Efficient ordered combinatorial semi-bandits for whole-page recommendation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.