Achieving Limited Adaptivity for Multinomial Logistic Bandits

Anonymous authors
Paper under double-blind review

Keywords: Multinomial Logistic Bandits, Limited Adaptivity, Batched Bandits, Contextual Bandits

Summary

Multinomial Logistic Bandits have recently attracted much attention due to their ability to model problems with multiple outcomes. In the multinomial model, each decision is associated with many possible outcomes, modeled using a multinomial logit function. Several recent works on multinomial logistic bandits have simultaneously achieved optimal regret and computational efficiency. However, motivated by real-world challenges and practicality, there is a need to develop algorithms with limited adaptivity, wherein we are allowed M policy updates only. To address these challenges, we present two algorithms, `B-MNL-CB` and `RS-MNL`, that operate in the batched and rarely-switching paradigms, respectively. The batched setting involves choosing the M policy update rounds at the start of the algorithm, while the rarely-switching setting can choose these M policy update rounds in an adaptive fashion. Our first algorithm, `B-MNL-CB` extends the notion of distributional optimal designs to the multinomial setting and achieves $\tilde{O}(\sqrt{T})$ regret assuming the contexts are generated stochastically when presented with $\Omega(\log \log T)$ update rounds. Our second algorithm, `RS-MNL` works with adversarially generated contexts and can achieve $\tilde{O}(\sqrt{T})$ regret with $\tilde{O}(\log T)$ policy updates. Further, diverse experiments demonstrate that our algorithms (with a fixed number of policy updates) are extremely competitive to several state-of-the-art baselines (which update their policy every round), showcasing the applicability of our algorithms in various practical scenarios.

Contribution(s)

1. We present an algorithm, `B-MNL-CB`, that achieves an optimal $O(\sqrt{T})$ regret with $\Omega(\log \log T)$ batches in the batched setting. Moreover, the leading term of the regret is independent of κ, an instance-dependent non-linearity parameter.
   Context: In the batched setting, the policy-update rounds are fixed. Gao et al. (2019b) showed that having $\Omega(\log \log T)$ batches is necessary to achieve the optimal minimax regret. Our algorithm, `B-MNL-CB`, uses the idea of distributional optimal designs, introduced in Ruan et al. (2021) and the scaling techniques used to learn the designs in Sawarni et al. (2024), and naturally extends the idea of distributional optimal designs to the multinomial logit setting. Achieving a $\kappa-$ independent regret is important because Amani & Thrampoulidis (2021) showed that κ scales exponentially in several instance parameters and hence, can increase the regret significantly.
2. We present a rarely-switching algorithm `RS-MNL`that achieves an optimal $O(\sqrt{T}$ regret (with a $\kappa-$free leading term) requiring $O(\log T)$ switches (policy updates) rounds
   Context: In the rarely-switching setting, the policy update (switch) rounds are adaptively chosen during the course of the algorithm. The need for the update is decided based on a switching criterion similar to the one in Abbasi-Yadkori et al. (2011). While the algorithm bears similarities to the rarely-switching algorithm presented in Sawarni et al. (2024), an alternate regret decomposition method allows us to get rid of the warm-up criterion, which helps reduce the number of switches from $O(\log^2 T)$ to $O(\log T)$. Further, we also get rid of the successive elimination method of choosing an arm in Sawarni et al. (2024) and replace it with the simpler UCB-maximization rule of Abbasi-Yadkori et al. (2011), resulting in a more efficient runtime for the algorithm.

Achieving Limited Adaptivity for Multinomial Logistic Bandits

Anonymous authors
Paper under double-blind review

Abstract

Multinomial Logistic Bandits have recently attracted much attention due to their ability to model problems with multiple outcomes. In the multinomial model, each decision is associated with many possible outcomes, modeled using a multinomial logit function. Several recent works on multinomial logistic bandits have simultaneously achieved optimal regret and computational efficiency. However, motivated by real-world challenges and practicality, there is a need to develop algorithms with limited adaptivity, wherein we are allowed M policy updates only. To address these challenges, we present two algorithms, `B-MNL-CB` and `RS-MNL`, that operate in the batched and rarely-switching paradigms, respectively. The batched setting involves choosing the M policy update rounds at the start of the algorithm, while the rarely-switching setting can choose these M policy update rounds in an adaptive fashion. Our first algorithm, `B-MNL-CB` extends the notion of distributional optimal designs to the multinomial setting and achieves $\tilde{O}(\sqrt{T})$ regret assuming the contexts are generated stochastically when presented with $\Omega(\log\log T)$ update rounds. Our second algorithm, `RS-MNL` works with adversarially generated contexts and can achieve $\tilde{O}(\sqrt{T})$ regret with $\tilde{O}(\log T)$ policy updates. Further, diverse experiments demonstrate that our algorithms (with a fixed number of policy updates) are extremely competitive to several state-of-the-art baselines (which update their policy every round), showcasing the applicability of our algorithms in various practical scenarios.

1 Introduction and Prior Works

Contextual Bandits help incorporate additional information that a learner may have with the standard Multi-Armed Bandit (MAB) setting. In this setting, at each round, the learner is presented with a set of arms and is expected to choose an arm. She is also presented with a context vector that helps guide the decisions she makes. For each decision, the learner receives a reward, which is generated using a hidden optimal parameter. The goal of the learner is to minimize her cumulative regret (or equivalently, maximize her cumulative reward), over a specified number of rounds T. Contextual Bandits have long been studied under various notions of reward models and settings. For instance, one of the simplest models is to assume that the expected reward is a linear function of the arms and the hidden parameter (Abbasi-Yadkori et al., 2011; Auer, 2003; Chu et al., 2011). This was later extended to non-linear settings such as the logistic (Faury et al., 2020; Abeille et al., 2021; Faury et al., 2022), generalized linear setting Filippi et al. (2010); Li et al. (2017), and the multinomial setting (Amani & Thrampoulidis, 2021; Zhang & Sugiyama, 2023). In this work, we specifically focus on the multinomial setting that can model problems with multiple outcomes, which makes this setting incredibly useful in the fields of machine and reinforcement learning, as well as, in real life.

Though significant progress has been made in designing algorithms for the contextual setting, the algorithms do not demonstrate a lot of applicability. There has been growing interest in constraining the budget available for algorithmic updates. This *limited adaptivity* setting is crucial in real-world applications, where frequent updates can hinder parallelism and large-scale deployment. Additionally, practical and computational constraints may make it infeasible to make policy updates at every time step. For example, in clinical trials (Group et al., 1997), the treatments made available to the patients cannot be changed with every patient. Thus, the updates are made after administering the treatment to a group of patients, observing the effects and outcomes, and then updating the treatment. We observe a similar tendency in online advertising and recommendations, where it is difficult to update the policy at each round due to resource constraints. A recent line of work (Ruan et al., 2021; Sawarni et al., 2024) has introduced algorithms for contextual bandits in the linear and generalized linear settings, respectively. They introduce algorithms for two different settings: the *batched* setting, wherein the policy update rounds are fixed at the start of the algorithm, and the *rarely-switching* algorithm, wherein the policy update rounds are decided in an adaptive fashion. Since multinomial logistic bandits are not generalized linear models, it is not clear if the algorithms developed in past works would apply in this setting. Hence, the major focus of this work is to develop algorithms with limited adaptivity for the multinomial setting. We now list our contributions:

1.1 Contributions

- We propose a new algorithm B-MNL-CB, which operates in the batched setting where the contexts are generated stochastically. The algorithm achieves $\tilde{O}(\sqrt{T})$ regret with high probability, with $\Omega(\log \log T)$ policy updates. In order to accommodate time-varying contexts, we adapt the recently introduced concept of distributional optimal designs (Ruan et al., 2021) to the multinomial logistic setting. This is done by introducing a new scaling technique to counter the non-linearity associated with the reward function. Note that the leading term of the regret bound is free of the instance-dependent non-linearity parameter κ, which can scale exponentially with the instance parameters (refer to Section 2 for more details).
- Our second algorithm, RS-MNL operates in the rarely-switching setting, where the contexts are generated adversarially. The algorithm achieves $\tilde{O}(\sqrt{T})$ regret while performing $\tilde{O}(\log T)$ policy updates, each determined by a simple switching criterion. Further, our algorithm does not require a warmup switching criterion, unlike the rarely-switching algorithm in Sawarni et al. (2024), which helps in reducing the number of switches from $\tilde{O}(\log^2 T)$ to $\tilde{O}(\log T)$.
- We conduct extensive experiments to demonstrate the performance of our rarely-switching algorithm RS-MNL. Across a wide range of randomly selected instances, our algorithm achieves regret comparable to, and often better than, several logistic and multinomial logistic state-of-the-art baseline algorithms. Our algorithm manages to do so with a limited number of policy updates as compared to the baselines, which perform an update at each time round. We also empirically validate that the number of switches made by our algorithm is $\tilde{O}(\log T)$, which is in agreement with our theoretical results.

1.2 Related works

Amani & Thrampoulidis (2021) were one of the first ones to deal with multinomial logistic setting. They proposed an algorithm that achieved a regret bound of $\tilde{O}(K\sqrt{\kappa T})$, where K is the number of outcomes and κ is the instance-dependent non-linearity parameter (defined in Section 2). This was further improved by Zhang & Sugiyama (2023), who proposed a computationally efficient algorithm with a regret bound of $\tilde{O}(\sqrt{T})$, thus achieving $\kappa-$free bounds (the leading term is free of κ). However, both of these algorithms face challenges in real-world deployment due to infrastructural and practical constraints associated with updating the policy at every round.

Thus, the limited adaptivity framework was introduced to combat this challenge, wherein the algorithm could only undergo a limited number of policy switches. This framework consists of two paradigms: the first being the *Batched* Setting, where the batch lengths are predetermined and was first studied by Gao et al. (2019a), who showed that $\Omega(\log \log T)$ batches are necessary to obtain optimal minimax regret. The second setting is the *Rarely Switching* Setting, first introduced by Abbasi-Yadkori et al. (2011), where batch lengths are determined adaptively, based on a switching criterion, such as the determinant doubling switching criteria used by Abbasi-Yadkori et al. (2011).

In the contextual setting, Ruan et al. (2021) used optimal designs to study the case where the arm sets themselves were generated stochastically, providing a bound of $\tilde{O}(d \log d\sqrt{T})$ for the batched setting. This idea was then extended to the generalized linear setting by Sawarni et al. (2024), who proposed algorithms that could achieve $\kappa-$free regret in both the batched and rarely-switching settings. However to the best of our knowledge, the limited adaptivity framework has not yet been explored in the multinomial case. We extend the results of Sawarni et al. (2024) and Ruan et al. (2021) to the multinomial setting in the batched setting while preserving the regret bound of Zhang & Sugiyama (2023) in the first-order term. In the rarely-switching setting, we further build upon the work of Abbasi-Yadkori et al. (2011) and Sawarni et al. (2024) to adapt it for the multinomial case, maintaining the regret bound of Zhang & Sugiyama (2023) while also improving computational efficiency by reducing the total number of switches.

2 Preliminaries

Notations: We denote all vectors with bold lower case letters, matrices with bold upper case letters, and sets with upper case calligraphic symbols. We write $\boldsymbol{M} \succcurlyeq 0$, if matrix $\boldsymbol{M}$ is positive semi-definite (p.s.d). For a p.s.d matrix $\boldsymbol{M}$, we define the norm of a vector $\boldsymbol{x}$ with respect to $\boldsymbol{M}$ as $||\boldsymbol{x}||_{\boldsymbol{M}} = \sqrt{\boldsymbol{x}^\intercal \boldsymbol{M}\boldsymbol{x}}$ and the spectral norm of $\boldsymbol{M}$ as $||\boldsymbol{M}||_2 = \sqrt{\lambda_{max}\left(\boldsymbol{M}^\intercal \boldsymbol{M}\right)}$ where $\lambda_{max}\left(\boldsymbol{M}\right)$ denotes the maximum eigenvalue of $\boldsymbol{M}$. We denote the set $\{1, \ldots, N\}$ as $[N]$. The standard Tensor or Kronecker Product between two vectors $\boldsymbol{a} = (a_1, \ldots, a_m)^\intercal$ and $\boldsymbol{b} = (b_1, \ldots, b_n)^\intercal$ is given by $\boldsymbol{a} \otimes \boldsymbol{b} = (a_1 b_1, \ldots, a_1 b_n, a_2 b_1, \ldots, a_2 b_n, \ldots, a_m b_1, \ldots, a_m b_n)^\intercal$. Finally, we use $\Delta(X)$ to denote the set of all probability distributions over X.

Multinomial Logistic Bandits: In the Multinomial Logistic Bandit Setting, at each round t, the learner is presented with a set of arms $\mathcal{X}_t$, and is expected to choose an arm $\boldsymbol{x}_t \in \mathcal{X}_t$. Based on the learner's choice, the environment provides an outcome $y_t \in [K] \cup \{0\}$[1]. While choosing the arm at round t, the learner can utilize all prior information, which can be encoded in the filtration $\mathcal{F}_t = \sigma\left(\mathcal{F}_0, \boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_{t-1}, y_{t-1}\right)$, where $\mathcal{F}_0$ represents any prior information the learner had before starting the algorithm. The probability distribution over these $K+1$ outcomes is modeled using a multinomial logit function as follows:

$$\mathbb{P}\left\{y_t = i \mid \boldsymbol{x}_t, \mathcal{F}_t\right\} = \begin{cases} \dfrac{\exp\left(\boldsymbol{x}_t^\intercal \boldsymbol{\theta}_i^*\right)}{1+\sum\limits_{j=1}^{K} \exp\left(\boldsymbol{x}_t^\intercal \boldsymbol{\theta}_j^*\right)} & 1 \le i \le K \\ \dfrac{1}{1+\sum\limits_{j=1}^{K} \exp\left(\boldsymbol{x}_t^\intercal \boldsymbol{\theta}_j^*\right)} & i = 0 \end{cases}$$

where $\boldsymbol{\theta}^\star = \left(\boldsymbol{\theta}_1^{\star\intercal}, \ldots, \boldsymbol{\theta}_K^{\star\intercal}\right)^\intercal \in \mathbb{R}^{dK}$ comprises the hidden optimal parameter vectors associated with each of the K outcomes. Based on the outcome y_t, the learner receives a reward $\rho_{y_t} \ge 0$. It is standard to set $\rho_0 = 0$. We assume that the reward vector $\rho = (\rho_1, \ldots, \rho_K)^\intercal$ is fixed and known. We assume that $||\boldsymbol{\theta}^\star||_2 \le S$, $||\boldsymbol{\rho}||_2 \le R$, and $||\boldsymbol{x}||_2 \le 1$, for all $\boldsymbol{x} \in \mathcal{X}_t$, where R and S are fixed and known beforehand. Note that when $K = 1$, the problem reduces to the binary logistic setting. For simplicity, we denote the probability of the i^{th} outcome $\mathbb{P}\left\{y_t = i \mid \boldsymbol{x}_t, \mathcal{F}_t\right\}$ as $z_i(\boldsymbol{x}_t, \boldsymbol{\theta}^\star)$ and denote the probability vector over the K outcomes as $\boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{\theta}^\star) = \left(z_1(\boldsymbol{x}_t, \boldsymbol{\theta}^\star), \ldots, z_K(\boldsymbol{x}_t, \boldsymbol{\theta}^\star)\right)^\intercal$. Then, it is easy to see that the expected reward of the learner at round t is given by $\boldsymbol{\rho}^\intercal \boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{\theta}^\star)$. The goal of the learner is to choose an arm $\boldsymbol{x}_t, t \in [T]$ so as to minimize her regret, which can have different formulations based on the problem setting:

[1] The outcome 0 indicates *no outcome*

1. Stochastic Contextual setting : In this setting, at each time step, the feasible action sets are sampled from the same (unknown) distribution $\mathcal{D}$. Thus, the learner wishes to minimize her expected cumulative regret which is given by

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T}\left[\max_{\boldsymbol{x}\in\mathcal{X}_t} \boldsymbol{\rho}^\top \boldsymbol{z}(\boldsymbol{x},\boldsymbol{\theta}^\star) - \boldsymbol{\rho}^\top \boldsymbol{z}(\boldsymbol{x}_t,\boldsymbol{\theta}^\star)\right]\right]$$

Here, the expectation is over the distribution of the arm set $\mathcal{D}$ and the randomness inherently present in the algorithm. In this setting, we assume that only M (fixed beforehand) policy updates can be made and the rounds at which these updates can happen need to be decided prior to starting the algorithm.

2. Adversarial Contextual setting : In this setting, there are no assumptions made on how the feature vectors of the arms are generated. Thus, allowed M policy updates, the algorithm can dynamically choose when to make the updates during the course of the algorithm and does not have to decide them in the beginning. These dynamic updates are based on a simple switching criterion similar to the one presented in Abbasi-Yadkori et al. (2011). In this setting, the learner wishes to minimize her cumulative regret given by

$$R(T) = \sum_{t=1}^{T}\left[\max_{\boldsymbol{x}\in\mathcal{X}_t} \boldsymbol{\rho}^\top \boldsymbol{z}(\boldsymbol{x},\boldsymbol{\theta}^\star) - \boldsymbol{\rho}^\top \boldsymbol{z}(\boldsymbol{x}_t,\boldsymbol{\theta}^\star)\right]$$

Discussion on the Instance-Dependent Non-Linearity Parameter κ: Several works on the binary logistic model and generalized linear model (Filippi et al., 2010; Faury et al., 2020) as well as the multinomial logistic model (Amani & Thrampoulidis, 2021; Zhang & Sugiyama, 2023) have mentioned the importance of an instance dependent, non-linearity parameter κ, and have stressed on the need to obtain regret guarantees independent of κ (at least in the leading term). In Section 2, Faury et al. (2020), it was highlighted that that κ can grow exponentially in the instance parameters such as S and therefore regret proportional to κ could be detrimental when these parameters are large. κ was first defined for the binary logistic reward model setting Filippi et al. (2010). A natural extension to the multinomial logit setting was recently proposed in Amani & Thrampoulidis (2021). We use the same definition as Amani & Thrampoulidis (2021), i.e.,

$$\kappa = \sup\left\{\frac{1}{\lambda_{min}(\boldsymbol{A}(\boldsymbol{x},\boldsymbol{\theta}))} : \boldsymbol{x}\in\mathcal{X}_1\cup\ldots\cup\mathcal{X}_T, \boldsymbol{\theta}\in\Theta\right\}$$

where $\boldsymbol{A}(\boldsymbol{x},\boldsymbol{\theta}) = \nabla\boldsymbol{z}(\boldsymbol{x},\boldsymbol{\theta}) = diag(\boldsymbol{z}(\boldsymbol{x},\boldsymbol{\theta})) - \boldsymbol{z}(\boldsymbol{x},\boldsymbol{\theta})\boldsymbol{z}(\boldsymbol{x},\boldsymbol{\theta})^\top$, is the gradient of the link function $\boldsymbol{z}$. In Section 3 of Amani & Thrampoulidis (2021), the authors show that κ in the multinomial setting also scales exponentially with the diameter of the parameter and action sets. We direct the reader to Section 3 of Amani & Thrampoulidis (2021) for a more elaborate discussion on the importance of κ in the multinomial setting.

Optimal Design policies: Optimal Experimental Designs are concerned with efficiently selecting the best data points so as to minimize the variance (or equivalently, maximize the information) of estimated parameters. For a set of points $\mathcal{X}$ and some distribution π defined on $\mathcal{X}$, The information matrix is defined as the inverse of the variance matrix $\mathbb{E}_{x\sim\pi}xx^\top$. Several criteria are used to maximize the information, some of which are A-Criterion (minimize trace of the information matrix), E-Criterion (maximize the minimum eigenvalue of the information matrix), and D-Criterion (maximize the determinant of the information matrix). One of the popular criteria used in bandit literature is the G-Optimal Design, which aims to minimize the maximum variance of the arms chosen.

Definition 2.1. G-Optimal Design: For a set $\mathcal{X}\subseteq\mathbb{R}^d$, the G-Optimal design $\pi_G(\mathcal{X})$ is the solution to the following optimization problem:

$$\min_{\pi\in\Delta(\mathcal{X})}\max_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_{V(\pi)^{-1}} \text{ where } \boldsymbol{V}(\pi) = \sum_{\boldsymbol{x}\in\mathcal{X}} \pi(\boldsymbol{x})\boldsymbol{x}\boldsymbol{x}^\top$$

The General Equivalence Theorem of G and D-Optimal designs (Kiefer & Wolfowitz, 1960; Lattimore & Szepesvári, 2020) establishes an equivalence between the G-Optimal and D-Optimal designs and show that the maximum variance is bounded above by d, the dimensionality of the arm sets. However, this notion of optimal designs can only be applied to fixed arm sets. In light of this, Ruan et al. (2021) introduced Distributional Optimal Designs for arm sets that are stochastically generated from an unknown distribution $\mathcal{D}$. In the linear setting, they show that using G-optimal designs directly for stochastically generated arm sets achieves $O(d\sqrt{\log(d)})$ regret (Theorem 2 Ruan et al. (2021)) and were able to reduce the regret by a factor of $O(\sqrt{d})$ using their novel distributional optimal design (Theorem 6, Ruan et al. (2021)). They define the distributional optimal design as a uniform distribution over the G-Optimal Design and the Mixed-Softmax policy which is defined below:

Definition 2.2. Softmax and Mixed-Softmax Policy:(Definition 3, Ruan et al. (2021)) The softmax policy $\pi^S_{\boldsymbol{M}}(\mathcal{X})$, for a fixed α, with respect to a positive semi-definite matrix $\boldsymbol{M}$ is defined as:

$$\pi^S_{\boldsymbol{M}}(\mathcal{X}) = \boldsymbol{x}_i \text{ if } i \sim softmax_\alpha(\boldsymbol{x}_1^\intercal \boldsymbol{M}\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N^\intercal \boldsymbol{M}\boldsymbol{x}_N)$$

where the softmax function (parametrized by α) is given by:

$$softmax_\alpha(s_1, \ldots s_N) = i \text{ with probability } \frac{s_i^\alpha}{\sum_{j=1}^N s_j^\alpha}$$

Now given a set $\mathcal{M} = (p_i, \boldsymbol{M}_i)_{i=1}^n$ such that $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$, the mixed-softmax policy is defined as follows:

$$\pi(\mathcal{X}) = \begin{cases} \pi_G(\mathcal{X}) & \text{with probability } \frac{1}{2} \\ \pi^S_{\boldsymbol{M}_i}(\mathcal{X}) & \text{with probability } \frac{p_i}{2} \end{cases} \tag{1}$$

The mixed-softmax policy is learned using the CoreLearning algorithm (Algorithm 3, Ruan et al. (2021)) which returns the distributional G-Optimal design given the set of context vectors S and a λ value. Sawarni et al. (2024) introduced the concept of scaled context vectors in order to use distributional optimal designs for generalized linear bandit models in the batched setting. In this paper, we propose an extension of this idea in the multinomial logit setting by introducing directionally scaled sets. These sets are used to learn the design policy utilized in the batched algorithm.

3 B-MNL-CB

Algorithm 1 B-MNL-CB

1: **Input:** M, $\boldsymbol{\rho}$, S, T
2: Initialize $\{\mathcal{T}_m\}_{m=1}^M$ as per 2, $\lambda = \sqrt{Kd\log T}$, and policy π_0 as G-OPTIMAL DESIGN
3: **for** batches $\beta = 1$ to M **do**
4:   **for** each round $t \in \mathcal{T}_\beta$ **do**
5:     Observe arm set X_t
6:     **for** $j = 1$ to $\beta - 1$ **do**
7:       Update arm set $\mathcal{X}_t \leftarrow UL_j(\mathcal{X}_t)$ (defined in 6)
8:     **end for**
9:     Sample $\boldsymbol{x}_t \sim \pi_{\beta-1}(\mathcal{X}_t)$ and obtain outcome y_t along with the corresponding reward ρ_{y_t}
10:   **end for**
11:   Equally divide $\mathcal{T}_\beta$ into two sets C and D
12:   Compute $\hat{\boldsymbol{\theta}}_\beta \leftarrow \arg\min \sum_{s\in C} \ell(\boldsymbol{\theta}, \boldsymbol{x}_s, y_s)$, $\boldsymbol{H}_\beta = \lambda \boldsymbol{I} + \sum_{s\in C} \frac{\boldsymbol{A}(\boldsymbol{x}_t, \hat{\boldsymbol{\theta}}_\beta) \otimes \boldsymbol{x}_t \boldsymbol{x}_t^\intercal}{B_\beta(\boldsymbol{x}_t)}$, and π_β using Algorithm 2 with the inputs $(\beta, \{\mathcal{X}_t\}_{t\in D})$
13: **end for**

In this section, we present our first algorithm B-MNL-CB,. We first introduce the algorithm and walk the reader through a step-by-step detailed explanation. We then mention a few salient remarks about our algorithm. Finally, we present the regret guarantee for our algorithm, provide a proof sketch for the same, and guide the reader to the full proof in the Appendix.

B-MNL-CB builds upon BATCHLINUCB-DG (Algorithm 5, Ruan et al. (2021)) and B-GLinCB (Algorithm 1, Sawarni et al. (2024)). This algorithm operates in the stochastic contextual setting (described in Section 2) within the batched paradigm. In this paradigm, the rounds at which the policy updates occur are fixed beforehand. We will refer to all the rounds between two consecutive policy updates as a *batch*. The horizon is divided into $M = O(\log\log T)$ disjoint batches denoted by $\{\mathcal{T}_\beta\}_{\beta=1}^{M}$, and the lengths of the batches are denoted by $\tau_\beta = |\mathcal{T}_\beta|$. Next, we describe the steps of Algorithm 1. The input to B-MNL-CB is the number of batches M, the fixed (known) reward vector $\boldsymbol{\rho}$, the known upper bound on $||\boldsymbol{\theta}^\star||_2$, i.e., S, and the total number of rounds T. In *Step 2*, we initialize λ to $\sqrt{Kd\log T}$ and set the batch lengths $\{\tau_\beta\}_{\beta=1}^{M}$ as per the rule mentioned in 2.

$$\tau_\beta = T^{1-2^{-\beta}} \tag{2}$$

In *Steps 4-13*, we iterate over all batches $\beta \in [M]$ and rounds $t \in \mathcal{T}_\beta$. During batch β and round $t \in \mathcal{T}_\beta$, first, in *Step 5*, we obtain the set of feasible arms $\mathcal{X}_t$ at round t. Then in *Steps 6-8*, we iterate over all the previous batches $j \in [\beta - 1]$ to prune $\mathcal{X}_t$ and retain only a subset of it via a *successive elimination* procedure described next.

3.1 Successive Elimination

For each prior batch $j \in [\beta - 1]$, we compute an upper confidence bound $UCB(j, \boldsymbol{x}, \lambda)$ and a lower confidence bound $LCB(j, \boldsymbol{x}, \lambda)$ as follows,

$$UCB(j, \boldsymbol{x}, \lambda) = \boldsymbol{\rho}^T \hat{\boldsymbol{\theta}}_j + \epsilon_1(j, \boldsymbol{x}, \lambda) + \epsilon_2(j, \boldsymbol{x}, \lambda) \tag{3}$$

$$LCB(j, \boldsymbol{x}, \lambda) = \boldsymbol{\rho}^T \hat{\boldsymbol{\theta}}_j - \epsilon_1(j, \boldsymbol{x}, \lambda) - \epsilon_2(j, \boldsymbol{x}, \lambda) \tag{4}$$

where the bonus terms $\epsilon_1(j, \boldsymbol{x}, \lambda)$ and $\epsilon_2(j, \boldsymbol{x}, \lambda)$ are defined as,

$$\epsilon_1(j, \boldsymbol{x}, \lambda) = \gamma(\lambda) \|\boldsymbol{H}_j^{-\frac{1}{2}} (\boldsymbol{I} \otimes \boldsymbol{x}) \boldsymbol{A}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_j) \boldsymbol{\rho}\|_2, \;\; \epsilon_2(j, \boldsymbol{x}, \lambda) = 3\gamma(\lambda)^2 \|\boldsymbol{\rho}\|_2 \|(\boldsymbol{I} \otimes \boldsymbol{x}^\intercal) \boldsymbol{H}_j^{-\frac{1}{2}}\|_2^2 \tag{5}$$

with $\widehat{\theta}_j$ and $\boldsymbol{H}_j$ being estimators (computed during *Steps 11,12* at the end of batch j) of the true parameter vector $\boldsymbol{\theta}^\star$ and an optimal batch-level Hessian matrix $\boldsymbol{H}_j^\star$. We provide more details on these in Section 3.2. In *Step 7*, for batch j, we eliminate a subset of $\mathcal{X}_t$ using the upper and lower confidence bounds just defined. In particular, we eliminate all $\boldsymbol{x} \in \mathcal{X}_t$ for which $UCB(j, \boldsymbol{x}, \lambda) \leq \max_{\boldsymbol{x}'} LCB(j, \boldsymbol{x}', \lambda)$. Thus, in *Step 7*, $\mathcal{X}_t$ is updated to $UL_j(\mathcal{X})$, defined as,

$$UL_j(\mathcal{X}) = \mathcal{X} \setminus \left\{ \boldsymbol{x} \in \mathcal{X} : UCB(j, \boldsymbol{x}, \lambda) \leq \max_{\boldsymbol{y} \in \mathcal{X}} LCB(j, \boldsymbol{y}, \lambda) \right\} \tag{6}$$

Following these successive eliminations for all prior batches $j \in [\beta - 1]$, in *Step 9*, we choose an arm $\boldsymbol{x}_t$ by sampling (from the remaining arms) according to a policy computed (using Algorithm 2) at the end of batch $\beta - 1$. The environment then provides the outcome y_t and the corresponding reward ρ_{y_t}. We provide details of the policy computation (Algorithm 2) in Section 3.3. After all rounds in batch β (i.e. $\mathcal{T}_\beta$) are complete, in *Step 11*, we partition these rounds equally into two sets C and D. The set C is used to define a batch-level Hessian matrix $\boldsymbol{H}_\beta^\star$ and to compute an estimator $\widehat{\theta}_\beta$ of $\theta^\star$ and a matrix $\boldsymbol{H}_\beta$ that estimates (a scaled version of) $\boldsymbol{H}_\beta^\star$ as follows.

3.2 Batch Level Hessian and Parameter Estimation

Using set C, for batch β, we define a batch level Hessian matrix $\boldsymbol{H}_\beta^\star = \lambda \boldsymbol{I} + \sum_{t \in C} \boldsymbol{A}(\boldsymbol{x}_t, \boldsymbol{\theta}^\star) \otimes \boldsymbol{x}_t \boldsymbol{x}_t^\intercal$. Since $\boldsymbol{\theta}^\star$ is unknown, we maintain an online proxy to estimate $\boldsymbol{H}_\beta^\star$ by calculating a scaled

Hessian matrix $\boldsymbol{H}_\beta = \lambda \boldsymbol{I} + \sum_{t \in C} \frac{\boldsymbol{A}(\boldsymbol{x}_t, \hat{\boldsymbol{\theta}}_\beta)}{B_\beta(\boldsymbol{x})} \otimes \boldsymbol{x}_t \boldsymbol{x}_t^\intercal$. Here, $B_\beta(\boldsymbol{x})$ is a normalizing factor which is obtained using the self-concordance properties of the link function and is given by:

$$B_\beta(\boldsymbol{x}) = \exp\left(\sqrt{6} \min\left\{\gamma(\lambda)\sqrt{\kappa}\, ||\boldsymbol{x}||_{\boldsymbol{V}_\beta^{-1}}\right\}, 2S\right) \tag{7}$$

where $\gamma(\lambda) = \mathcal{O}(\sqrt{Kd \log T})$ is the confidence radius for the permissible set of $\boldsymbol{\theta}$ and $\boldsymbol{V}_\beta$ is the design matrix given by $\boldsymbol{V}_\beta = \lambda \boldsymbol{I} + \sum_{t \in C} \boldsymbol{x}_t \boldsymbol{x}_t^\intercal$. Using the self-concordance results, we can show that $\boldsymbol{H}_\beta \preccurlyeq \boldsymbol{H}_\beta^\star$. We also use the set C to update the estimator $\hat{\boldsymbol{\theta}}_\beta$, which is done by minimizing the negative log likelihood $\sum\limits_{t \in C} \ell(\boldsymbol{\theta}, \boldsymbol{x}_t, y_t)$, where $\ell(\boldsymbol{\theta}, \boldsymbol{x}, y)$ is defined as,

$$\ell(\boldsymbol{\theta}, \boldsymbol{x}, y) = -\sum_{i=1}^{K} \mathbb{1}\{y = i\} \log \frac{1}{z_i(\boldsymbol{x}, \boldsymbol{\theta})} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 \tag{8}$$

Next, we explain how the policy is updated to π_β at the end of batch β using the rounds in set D.

3.3 Policy calculation

Algorithm 2 Distributional Optimal Design for MNL bandits

1: **Input** Batch β and collection of arm sets $\mathcal{S} = \{\mathcal{X}_t : t \in D\}$
2: Create the sets $\{F_i(\mathcal{S}, \beta)\}_{i=1}^K$ as defined in Equation 9
3: Compute the distributional optimal design policy π_i for each of the sets $F_i(\mathcal{S}, \beta)$
4: Compute the distributional optimal design policy π_0 for the set $\mathcal{S}$
5: **Return** $\pi = \frac{1}{K+1} \sum\limits_{i=0}^{K} \pi_i$

To compute our final policy, we utilize distributional optimal design (See Section 2) introduced in Ruan et al. (2021). Recently, Sawarni et al. (2024) used distributional optimal designs to develop limited adaptivity algorithms for stochastic contextual bandits with generalized linear reward models. A key step in their algorithm (Step 13 and Equation 4, Algorithm 1 in Sawarni et al. (2024)) involves scaling the set of arm feature vectors (post-sequential elimination) using the derivative of the link function and a suitable normalization factor. Generalizing this idea to the MNL setting results in a matrix $\tilde{\boldsymbol{X}} = \frac{\boldsymbol{A}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_t)^{\frac{1}{2}}}{B_\beta(\boldsymbol{x})} \otimes \boldsymbol{x}$. Since the notion of distributional optimal designs introduced in Ruan et al. (2021) and used by Sawarni et al. (2024), applies only to vectors, in Algorithm 2, we construct several sets of vectors from $\tilde{\boldsymbol{X}}$ and learn the optimal design for these sets.

In *Step 12* of Algorithm 1, we invoke this algorithm (Algorithm 2) with inputs as the batch number β and the collection $\mathcal{S}$ of all the pruned arm sets $\mathcal{X}_t$ (*Step 7*, Algorithm 1) for rounds $t \in D$, i.e. $\mathcal{S} = \{\mathcal{X}_t : t \in D\}$. We then create K different sets $F_i(\mathcal{S}, \beta)$ ($i \in [K]$), which comprises of the arms in each arm set scaled by the i^{th} column of the gradient matrix. In particular,

$$F_i(\mathcal{S}, \beta) = \left\{\left\{\frac{\boldsymbol{A}(\boldsymbol{x}, \widehat{\boldsymbol{\theta}}_\beta)^{\frac{1}{2}}}{\sqrt{B_\beta(\boldsymbol{x})}} \boldsymbol{e}_i \otimes \boldsymbol{x} : \boldsymbol{x} \in \mathcal{X}\right\} : \mathcal{X} \in \mathcal{S}\right\} \tag{9}$$

where $\boldsymbol{e}_i \in \mathbb{R}^K$ is the i^{th} standard basis vector. We calculate the distributional optimal design for each of the sets $F_i(\mathcal{S}, \beta)$ using Algorithm 2 in Ruan et al. (2021). In such a case, it is easy to see that calculating the distributional optimal design over $\tilde{\boldsymbol{X}}$ can be done by calculating the distributional optimal designs for each of the sets $F_i(\mathcal{S}, \beta)$. We provide the proof for the same in Section 8.2. We also calculate the distributional optimal design over $\mathcal{S}$. Finally, the policy returned is a convex combination (in this case, a uniform combination) over these $K + 1$ designs that were calculated.

This completes our explanation of Algorithm 1. We provide a regret guarantee in Theorem 3.4.

Remark 3.1. Directly borrowing the scaling techniques introduced in Sawarni et al. (2024) for learning distributional optimal designs in the multinomial setting results in the creation of a scaled matrix. Since the notion of distributional optimal design introduced in Ruan et al. (2021) applies only to vectors, Algorithm 2 scales the original context vectors into K different sets and then learns the optimal designs for each of them.

Remark 3.2. Sawarni et al. (2024) introduces a warm-up round whose length is $O(\kappa^{1/3})$. Since κ can scale exponentially with several instance-dependent parameters, the warm-up round can result in a long exploration phase. Using the regret decomposition in Zhang & Sugiyama (2023), we can eliminate the dependence on κ, resulting in $\kappa-$free batch lengths, including the length of the warm-up round.

Remark 3.3. While Zhang & Sugiyama (2023) introduced a novel method of regret decomposition into the error terms (refer 5), using the same decomposition in the limited adaptivity setting is not straightforward. Hence, with some additional insights, we incorporate their method into the batched setting while being able to match the leading term of their regret bound.

Theorem 3.4. *Let the number of batches $M = O(\log\log T)$, then, with a probability at least $1-\frac{1}{T^2}$, Algorithm 1 achieves a final regret bound of $R_T \leq (R_1 + R_2)$ where*

$$R_1 = \tilde{O}\left(RK^{\frac{5}{2}}d\sqrt{T\log(Kd)}\right)$$

$$R_2 = \tilde{O}\left(\frac{RK^4d^2T^{\frac{1}{4}}}{S}\left(e^{3S}\log(Kd) + \sqrt{\kappa}d\right)\right)$$

Proof Sketch:

We know that the expected regret during batch $\beta + 1$ is given by:

$$R_{\beta+1} = \mathbb{E}\left[\sum_{t\in\beta}\boldsymbol{\rho}^{\intercal}\left(\boldsymbol{z}(\boldsymbol{x}_t^{\star}, \boldsymbol{\theta}^{\star}) - \boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{\theta}^{\star})\right)\right]$$

where $\boldsymbol{x}_t^{\star} = \arg\max_{\boldsymbol{x}\in\mathcal{X}_t}\boldsymbol{\rho}^{\intercal}\boldsymbol{z}(\boldsymbol{x}, \boldsymbol{\theta}^{\star})$ is the best arm at round t and the expectation is taken over the distribution of the arm set $\mathcal{D}$. Using ideas similar to Zhang & Sugiyama (2023), we can decompose the regret into

$$R(T) \leq 4\sum_{t\in\beta}\left\{\mathbb{E}\left[\max_{\boldsymbol{x}\in\mathcal{X}_t}\epsilon_1(\beta, \boldsymbol{x}, \lambda)\right] + \mathbb{E}\left[\max_{\boldsymbol{x}\in\mathcal{X}_t}\epsilon_2(\beta, \boldsymbol{x}, \lambda)\right]\right\}$$

where $\epsilon_1(\beta, \boldsymbol{x}, \lambda)$ and $\epsilon_2(\beta, \boldsymbol{x}, \lambda)$ are as defined in 5. Now, we bound each of the terms separately using the new idea of distributional Optimal Designs introduced in Algorithm 2.

Directly extending the ideas of Ruan et al. (2021) and Sawarni et al. (2024) to construct the distributional optimal designs results in an attempt to learn the design for matrices $\tilde{\boldsymbol{X}}_\beta = \frac{\boldsymbol{A}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\beta)^{\frac{1}{2}}}{B_\beta(\boldsymbol{x})} \otimes \boldsymbol{x}$. Hence, we create K different sets $F_i(\mathcal{X})$ for all $i \in [K]$ (defined in 9), such that

$$\tilde{\boldsymbol{X}}_\beta\tilde{\boldsymbol{X}}_\beta^{\intercal} = \sum_{i=1}^{K}\left\{\frac{\boldsymbol{A}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\beta)^{\frac{1}{2}}}{\sqrt{B_\beta(\boldsymbol{x})}}\boldsymbol{e}_i \otimes \boldsymbol{x}\right\}\left\{\frac{\boldsymbol{A}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\beta)^{\frac{1}{2}}}{\sqrt{B_\beta(\boldsymbol{x})}}\boldsymbol{e}_i \otimes \boldsymbol{x}\right\}^T$$

Thus, learning the optimal design over $\tilde{\boldsymbol{X}}$ is equivalent to creating a convex combination of the designs learned over $F_i(\mathcal{X})$ for all $i \in [K]$. This gives us a way of bounding the scaled Hessian matrix $\boldsymbol{H}_\beta$ by the scaled Hessian matrices $\boldsymbol{H}_\beta^i$ constructed over $F_i(\mathcal{X})$ for all $i \in [K]$. We then use methods similar to Sawarni et al. (2024) and Ruan et al. (2021) to obtain the bound on the regret for the $\beta+1$ batch as:

$$R_{\beta+1} \leq CRK\gamma^2(\lambda)\sqrt{\kappa}d \left\{ \frac{e^{3S}K^{3/2}}{S} \sqrt{\log Kd \log d} + C\sqrt{\kappa}d \right\} \left(\frac{\tau_{\beta+1}}{\tau_\beta} \right) + CR\gamma(\lambda)K^2\sqrt{d \log Kd} \left(\frac{\tau_{\beta+1}}{\sqrt{\tau_\beta}} \right)$$

Finally, using the batch lengths defined in 2 and summing over all the M batches completes the proof. For the sake of brevity, we provide the complete proof in Section 8.

4 `RS-MNL`

Algorithm 3 RS-MNL

1: **Inputs:** $\boldsymbol{\rho}, S, T$
2: **Initialize:** $\boldsymbol{H}_1 = \lambda \boldsymbol{I}$, $\tau = 1$, $\lambda := \frac{d \log(T/\delta)}{R^2}$, $\gamma := 25RS$
3: **for** $t = 1, \ldots, T$ **do**
4: Observe arm set $\mathcal{X}_t$
5: **if** $\det(\boldsymbol{H}_t) > 2\det(\boldsymbol{H}_\tau)$ **then**
6: Set $\tau = t$
7: Update $\hat{\boldsymbol{\theta}}_\tau \leftarrow \arg\min_{\boldsymbol{\theta}} \sum_{s\in[t-1]} \ell(\boldsymbol{\theta}, \boldsymbol{x}_s, y_s)$ and $\boldsymbol{H}_t = \sum_{s\in[t-1]} \frac{\boldsymbol{A}(\boldsymbol{x}_s, \hat{\boldsymbol{\theta}}_\tau)}{B_\tau(\boldsymbol{x}_s)} \otimes \boldsymbol{x}_s\boldsymbol{x}_s^\intercal + \lambda \boldsymbol{I}_{Kd}$
8: **end if**
9: Select $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}\in\mathcal{X}_t} UCB(t, \tau, \boldsymbol{x})$, observe y_t, and update $\boldsymbol{H}_{t+1} \leftarrow \boldsymbol{H}_t + \frac{\boldsymbol{A}(\boldsymbol{x}_t, \hat{\boldsymbol{\theta}}_\tau)}{B_\tau(\boldsymbol{x}_t)} \otimes \boldsymbol{x}_t\boldsymbol{x}_t^\intercal$
10: **end for**

In this section, we present our second algorithm `RS-MNL`. We introduce the algorithm and explain the workings in a step-by-step fashion. We then mention a few salient remarks about our algorithm. We conclude with the regret guarantee of our algorithm, a proof sketch for the same, and for the sake of brevity, we provide the complete proof in the Appendix.

Our second algorithm, `RS-MNL` (Algorithm 3) operates in the Adversarial Contextual setting. In this setting, there are no assumptions on the generation of the feature vectors. `RS-MNL` also limits the number of policy updates in a rarely-switching fashion, i.e, the rounds where these updates are made are decided dynamically, based on a simple switching criterion, similar to the one used in Abbasi-Yadkori et al. (2011). While the algorithm is based on `RS-GLinCB` in Sawarni et al. (2024), a unique regret decomposition method allows for the removal of the warmup criterion, in turn, helping in the reduction in the number of switches made by the algorithm from $O(\log^2 T)$ to $O(\log T)$. Further, we successfully remove the successive eliminations based on the previous confidence regions and replace the idea with the maximization of the Upper Confidence Bound of each arm. The inputs to the algorithm are $\boldsymbol{\rho}$, the fixed and known reward vector, S, the fixed and known upper bound on $\|\boldsymbol{\theta}\|_2$, and T, the number of rounds for which the algorithm is played. In *Step 2* we initialize the scaled Hessian matrix $\boldsymbol{H}_1$ to $\boldsymbol{I}$, λ to $\frac{d \log(T/\delta)}{R^2}$, and γ to $25RS$. Next, at every time round $t \in [T]$, we receive the arm set $\mathcal{X}_t$ in *Step 4*. During *Steps 5-8*, we check if the switching condition is met and update the policy accordingly.

4.1 Switching Criterion and Policy Update:

We use τ to keep track of the time step at which the policy was last changed during some round t. In *Step 5*, we evaluate if the determinant of the scaled Hessian matrix $\boldsymbol{H}_t = \lambda \boldsymbol{I} + \sum_{s\in[t-1]} \frac{\boldsymbol{A}(\boldsymbol{x}_s, \hat{\boldsymbol{\theta}}_\tau)}{B(\boldsymbol{x}_s)} \otimes \boldsymbol{x}_s\boldsymbol{x}_s^\intercal$ has increased by a constant factor (in this case, 2) as compared to $\boldsymbol{H}_\tau$. In case it is triggered, at *Step 6* we set $\tau = t$ since t is now the most recent switching round. We then compute $\hat{\boldsymbol{\theta}}_\tau$ by minimizing the negative log likelihood $\sum_{s\in[t-1]} \ell(\boldsymbol{\theta}, \boldsymbol{x}_s, y_s)$ (see 8 for definition of $\ell(\boldsymbol{\theta}, \boldsymbol{x}_s, y_s)$)

over all previous rounds $s \in [t-1]$, and recompute the matrix $\boldsymbol{H}_t$ with respect to the newly calculated $\hat{\boldsymbol{\theta}}_\tau$ (*Step 7*). The switching criterion is similar to the one used in Abbasi-Yadkori et al. (2011) and helps to reduce the number of policy updates to $O(\log T)$.

4.2 Arm Selection:

Next, in *Step 9*, we determine the arm $\boldsymbol{x}_t$ to be played based on the Upper Confidence Bound (UCB). The upper confidence bound $UCB(t, \tau, \boldsymbol{x})$ for an arm $\boldsymbol{x} \in \mathcal{X}_t$ with respect to the previous switching round $\tau(\leq t)$ is defined as:

$$UCB(t, \tau, \boldsymbol{x}) = \boldsymbol{\rho}^T \hat{\boldsymbol{\theta}}_\tau + \epsilon_1(t, \tau, \boldsymbol{x}) + \epsilon_2(t, \tau, \boldsymbol{x}) \tag{10}$$

where the error terms $\epsilon_1(t, \tau, \boldsymbol{x})$ and $\epsilon_2(t, \tau, \boldsymbol{x})$ are defined as:

$$\epsilon_1(t, \tau, \boldsymbol{x}) = \sqrt{2}\gamma(\delta)\|\boldsymbol{H}_t^{-\frac{1}{2}}(\boldsymbol{I} \otimes \boldsymbol{x})\boldsymbol{A}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\tau)\boldsymbol{\rho}\|_2, \;\; \epsilon_2(t, \tau, \boldsymbol{x}) = 6R\gamma(\delta)^2\|(\boldsymbol{I} \otimes \boldsymbol{x}^\intercal)\boldsymbol{H}_t^{-\frac{1}{2}}\|_2^2 \tag{11}$$

We then obtain the outcome y_t, which is sampled from $\boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{\theta}^\star)$, and receives the corresponding reward ρ_{y_t}. The algorithm then updates the scaled Hessian matrix $\boldsymbol{H}_{t+1}$. In Theorem 4.3, we provide the regret guarantee for `RS-MNL`.

Remark 4.1. The goal of a rarely-switching algorithm is to reduce the number of policy updates (switches) that are done. Our algorithm successfully reduces the number of switches from $\mathcal{O}(\log^2 T)$ to $O(\log T)$ due to the removal of the warm-up switching criterion. Additionally, the number of switches is independent of κ.

Remark 4.2. Similar to the batched setting, using the regret decomposition method introduced in Zhang & Sugiyama (2023) in the rarely-switching paradigm is non-trivial. We manage to extend their results to match the leading term of their regret bound while performing a switch $O(\log T)$ times.

Theorem 4.3. *With probability $\geq 1-\delta$, where $\delta \in (0,1)$, Algorithm 3 achieves the following regret:*

$$R(T) \leq CRK^{\frac{3}{2}}Sd\log\left(\frac{T}{\delta}\right)\sqrt{T} + CRd^2S^{\frac{1}{2}}e^{3S}\kappa^{\frac{1}{4}}K^2\log^2\left(\frac{1}{\delta}\right)T^{\frac{1}{4}} + CRS^2d^2\log^2\left(\frac{T}{\delta}\right)K^3\kappa e^{2S}$$

Proof Sketch:

The expression for total regret is given by

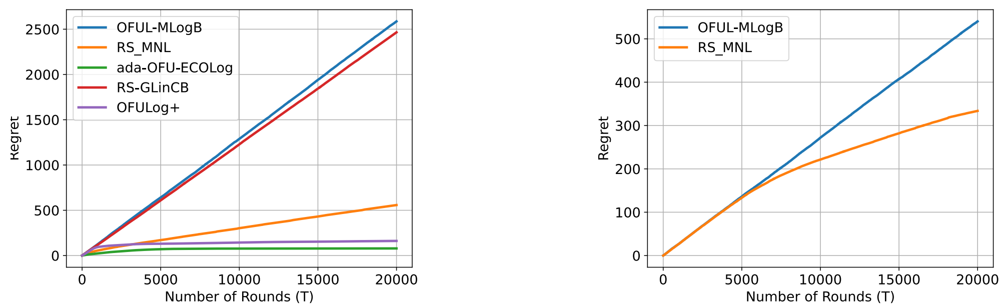
$$R(T) = \sum_{t=1}^{T} \boldsymbol{\rho}^\intercal \left(\boldsymbol{z}(\boldsymbol{x}_t^\star, \boldsymbol{\theta}^\star) - \boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{\theta}^\star)\right)$$

where $\boldsymbol{x}_t^\star = \arg\max_{\boldsymbol{x} \in \mathcal{X}_t} \boldsymbol{\rho}^\intercal \boldsymbol{z}(\boldsymbol{x}, \boldsymbol{\theta}^\star)$ is the best arm at any given round t. Using methods similar to Zhang & Sugiyama (2023), we can upper bound the regret as

$$R(T) \leq 2\sum_{t=1}^{T}\{\epsilon_1(t, \tau, \boldsymbol{x}_t) + \epsilon_2(t, \tau, \boldsymbol{x}_t)\}$$

where $\epsilon_1(t, \tau, \boldsymbol{x}_t)$ and $\epsilon_2(t, \tau, \boldsymbol{x}_t)$ are as defined in 11. We now wish to upper bound both the terms separately.

Bounding $\epsilon_1(t, \tau, \boldsymbol{x}_t)$ using a similar switching criterion in Abbasi-Yadkori et al. (2011) alongside the selection rule in our algorithm can result in an exponential dependency in S, which was circumvented by Sawarni et al. (2024) using a warm-up criterion. However, this warm-up criterion increases the number of switches from $O(\log T)$ to $O(\log^2 T)$. It also slows down the algorithm



(a) Regret vs. T: Logistic ($K = 1$) Setting

(b) Regret vs. T: $K = 3$

Figure 1

due to the successive eliminations done at each round (similar to the ones in Algorithm 1). Our algorithm gets rid of the exponential dependency and the warm-up criterion by further decomposing $\epsilon_1(t, \tau, \boldsymbol{x}_t)$ in an alternate manner, resulting in an improved runtime as well as $O(\log T)$ switches.

We bound both $\epsilon_1(t, \tau, \boldsymbol{x}_t)$ and $\epsilon_2(t, \tau, \boldsymbol{x}_t)$ using an analysis similar to the one used for Theorem 8.9, where we attempt to upper bound the scaled Hessian matrix $\boldsymbol{H}_t$ using the scaled Hessian matrices calculated over the K different scaled sets introduced in 2 even though these sets do not explicitly appear anywhere in the algorithm. Combining the bounds on each of the error terms finishes the proof. For the sake of brevity, we provide the complete proof in Section 9 of the Appendix.

5 Experiments

Experiment 1 ($R(T)$ vs. T for the Logistic Setting): In this experiment, we compare our algorithm RS-MNL to several state-of-the-art contextual logistic bandit algorithms. We set the number of outcomes K to 1, which reduces the problem to the logistic setting, where we compare our algorithm to ada-OFU-ECOLog (Algorithm 2, Faury et al. (2022)), RS-GLinCB (Algorithm 2, Sawarni et al. (2024), OFUL-MLogB (Algorithm 2, Zhang & Sugiyama (2023)), and OFULog+ (Algorithm 1, Lee et al. (2024)). The arm set $\mathcal{X}$ is randomly sampled from $[-1, 1]^3$ and the number of arms $|\mathcal{X}|$ is set to 10. We simulate $\boldsymbol{\theta}^\star$ from $[-1, 1]^3$ and normalize it to a unit vector. We run all the algorithms for $T = 20000$ rounds and plot our results in Figure 1a. We can see that RS-MNL incurs lower regret than RS-GLinCB and OFUL-MLogB, while performing slightly worse than ada-OFU-ECOLog and OFULog+.

Experiment 2 ($R(T)$ vs. T for $K = 3$): In this experiment, we compare our algorithm RS-MNL to OFUL-MLogB, the only algorithm to the best of our knowledge that achieves an optimal ($\kappa-$free) regret while being computationally efficient. The arm set $\mathcal{X}$ is randomly sampled from $[-1, 1]^3$ and the number of arms $|\mathcal{X}|$ are fixed to 10. We simulate $\boldsymbol{\theta}^\star$ from $[-1, 1]^9$ (since $\boldsymbol{\theta}^\star \in \mathbb{R}^{Kd}$ and normalize it to a unit vector. We run the algorithm for $T = 20000$ rounds and plot our results in Figure 1b. We can see that RS-MNL incurs lower regret than OFUL-MLogB.

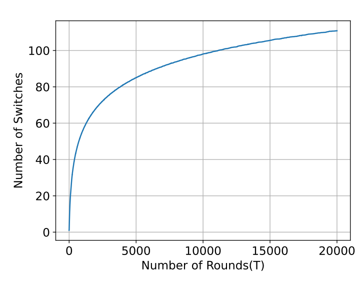


Figure 2: Switches vs. T

Experiment 3 (Number of Switches vs. T): In this experiment, we plot the number of switches RS-MNL makes as a function of the number of rounds T. We assume that the instances are simulated in the same manner as **Experiment 1** and **Experiment 2**. We vary the number of outcomes K in the set $\{1, \ldots, 6\}$ and average the number of switches made at each round $t \in [T]$ over 5 different instances for each K. The results are shown in Figure 2, and we can see that the number of switches exhibits a logarithmic dependence with T. This is in agreement with Lemma 9.13, where we show that RS-MNL switches $O(\log T)$ times.

6 Conclusions and Future Work

In this paper, we present two algorithms B-MNL-CB and RS-MNL, for the multinomial logit setting in the batched and rarely-switching paradigms, respectively. The batched setting involves fixing the policy update rounds at the start of the algorithm, while the rarely switching setting chooses the policy update rounds adaptively. Our first algorithm, B-MNL-CB manages to extend the notion of distributional optimal designs to the multinomial logit setting while being able to achieve an optimal regret of $O(\sqrt{T})$ in $\Omega(\log\log T)$ batches. Our second algorithm, RS-MNL, builds upon the previous rarely-switching algorithm in Sawarni et al. (2024) and obtains an optimal regret of $O(\sqrt{T})$ while being able to reduce the number of switches to $O(\log T)$ using alternate ways of regret decomposition. The regret of our algorithms scales with the number of outcomes K as K^4 and K^3 respectively, which can be detrimental for problems with a large number of outcomes. We believe that this dependence on K can be further improved, which is an interesting future work.

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24 (NeurIPS)*, pp. 2312–2320, 2011.

Marc Abeille, Louis Faury, and Clement Calauzenes. Instance-wise minimax-optimal algorithms for logistic bandits. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3691–3699. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/abeille21a.html.

Sanae Amani and Christos Thrampoulidis. Ucb-based algorithms for multinomial logistic regression bandits, 2021. URL https://arxiv.org/abs/2103.11489.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3(null):397–422, March 2003. ISSN 1532-4435.

Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 208–214, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/chu11a.html.

Louis Faury, Marc Abeille, Clément Calauzènes, and Olivier Fercoq. Improved optimistic algorithms for logistic bandits, 2020. URL https://arxiv.org/abs/2002.07530.

Louis Faury, Marc Abeille, Kwang-Sung Jun, and Clément Calauzènes. Jointly efficient and optimal algorithms for logistic bandits, 2022. URL https://arxiv.org/abs/2201.01985.

Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/c2626d850c80ea07e7511bbae4c76f4b-Paper.pdf.

Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. Batched multi-armed bandits problem. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 503–513. Curran Associates, Inc., 2019a.

Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. Batched multi-armed bandits problem, 2019b. URL https://arxiv.org/abs/1904.01763.

International Stroke Trial Collaborative Group et al. The international stroke trial (ist): a randomised trial of aspirin, subcutaneous heparin, both, or neither among 19 435 patients with acute ischaemic stroke. *The Lancet*, 349(9065):1569–1581, 1997.

Osama A. Hanna, Lin F. Yang, and Christina Fragouli. Efficient batched algorithm for contextual linear bandits with large action space via soft elimination. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

J. Kiefer and J. Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960. DOI: 10.4153/CJM-1960-030-4.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Junghyun Lee, Se-Young Yun, and Kwang-Sung Jun. Improved regret bounds of (multinomial) logistic bandits via regret-to-confidence-set conversion, 2024. URL https://arxiv.org/abs/2310.18554.

L. Li, Y. Lu, and D. Zhou. Provably optimal algorithms for generalized linear contextual bandits. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2071–2080. JMLR.org, 2017.

Yufei Ruan, Jiaqi Yang, and Yuan Zhou. Linear bandits with limited adaptivity and learning distributional optimal design, 2021. URL https://arxiv.org/abs/2007.01980.

Ayush Sawarni, Nirjhar Das, Siddharth Barman, and Gaurav Sinha. Generalized linear bandits with limited adaptivity, 2024. URL https://arxiv.org/abs/2404.06831.

Yu-Jie Zhang and Masashi Sugiyama. Online (multinomial) logistic bandit: Improved regret and constant computation cost. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 29741–29782. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5ef04392708bb2340cb9b7da41225660-Paper-Conference.pdf.