Provably Efficient Reinforcement Learning with Linear Function Approximation under Adaptivity Constraints

Tianhao Wang*
Department of Statistics and Data Science
Yale University
New Haven, CT 06511
tianhao.wang@yale.edu

Dongruo Zhou*
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
drzhou@cs.ucla.edu

Quanquan Gu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
qgu@cs.ucla.edu

Abstract

We study reinforcement learning (RL) with linear function approximation under the adaptivity constraint. We consider two popular limited adaptivity models: the batch learning model and the rare policy switch model, and propose two efficient online RL algorithms for episodic linear Markov decision processes, where the transition probability and the reward function can be represented as a linear function of some known feature mapping. In specific, for the batch learning model, our proposed LSVI-UCB-Batch algorithm achieves an $\widetilde{O}(\sqrt{d^3H^3T} + dHT/B)$ regret, where d is the dimension of the feature mapping, H is the episode length, T is the number of interactions and B is the number of batches. Our result suggests that it suffices to use only $\sqrt{T/dH}$ batches to obtain $\widetilde{O}(\sqrt{d^3H^3T})$ regret. For the rare policy switch model, our proposed LSVI-UCB-RareSwitch algorithm enjoys an $\widetilde{O}(\sqrt{d^3H^3T[1+T/(dH)]^{dH/B}})$ regret, which implies that $dH\log T$ policy switches suffice to obtain the $\widetilde{O}(\sqrt{d^3H^3T})$ regret. Our algorithms achieve the same regret as the LSVI-UCB algorithm (Jin et al., 2020), yet with a substantially smaller amount of adaptivity. We also establish a lower bound for the batch learning model, which suggests that the dependency on B in our regret bound is tight.

1 Introduction

Real-world reinforcement learning (RL) applications often come with possibly infinite state and action space, and in such a situation classical RL algorithms developed in the tabular setting are not applicable anymore. A popular approach to overcoming this issue is by applying function approximation techniques to the underlying structures of the Markov decision processes (MDPs). For example, one can assume that the transition probability and the reward are linear functions of a known feature mapping $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, and d is the dimension of the embedding. This gives rise to the so-called linear MDP model (Yang and Wang, 2019; Jin et al., 2020). Assuming access to a generative model, efficient algorithms under

*Equal contribution

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

this setting have been proposed by Yang and Wang (2019) and Lattimore et al. (2020). For online finite-horizon episodic linear MDPs, Jin et al. (2020) proposed an LSVI-UCB algorithm that achieves $\widetilde{O}(\sqrt{d^3H^3T})$ regret, where H is the planning horizon (i.e., length of each episode) and T is the number of interactions.

However, all the aforementioned algorithms require the agent to update the policy in every episode. In practice, it is often unrealistic to frequently switch the policy in the face of big data, limited computing resources as well as inevitable switching costs. Thus one may want to batch the data stream and update the policy at the end of each period. For example, in clinical trials, each phase (batch) of the trial amounts to applying a medical treatment to a batch of patients in parallel. The outcomes of the treatment are not observed until the end of the phase and will be subsequently used to design experiments for the next phase. Choosing the appropriate number and sizes of the batches is crucial to achieving nearly optimal efficiency for the clinical trial. This gives rise to the limited adaptivity setting, which has been extensively studied in many online learning problems including prediction-from-experts (PFE) (Kalai and Vempala, 2005; Cesa-Bianchi et al., 2013), multi-armed bandits (MAB) (Arora et al., 2012; Cesa-Bianchi et al., 2013) and online convex optimization (Jaghargh et al., 2019; Chen et al., 2020), to mention a few. Nevertheless, in the RL setting, learning with limited adaptivity is relatively less studied. Bai et al. (2019) introduced two notions of adaptivity in RL, *local switching cost* and *global switching cost*, that are defined as follows

$$N_{\text{local}} = \sum_{k=1}^{K-1}\sum_{h=1}^{H}\sum_{s\in\mathcal{S}} \mathbb{1}\{\pi_h^k(s) \neq \pi_h^{k+1}(s)\} \quad \text{and} \quad N_{\text{global}} = \sum_{k=1}^{K-1} \mathbb{1}\{\pi^k \neq \pi^{k+1}\}, \tag{1.1}$$

where $\pi^k = \{\pi_h^k : \mathcal{S} \to \mathcal{A}\}_{h\in[H]}$ is the policy for the k-th episode of the MDP, $\pi^k \neq \pi^{k+1}$ means that there exists some $(h, s) \in [H] \times \mathcal{S}$ such that $\pi_h^k(s) \neq \pi_h^{k+1}(s)$, and K is the number of episodes. Then they proposed a Q-learning method with UCB2H exploration that achieves $\widetilde{O}(\sqrt{H^3SAT})$ regret with $O(H^3SA\log(T/(AH)))$ local switching cost for tabular MDPs, but they did not provide tight bounds on the global switching cost.

In this paper, based on the above motivation, we aim to develop online RL algorithms with linear function approximation under adaptivity constraints. In detail, we consider time-inhomogeneous[2] episodic linear MDPs (Jin et al., 2020) where both the transition probability and the reward function are unknown to the agent. In terms of the limited adaptivity imposed on the agent, we consider two scenarios that have been previously studied in the online learning literature (Perchet et al., 2016; Abbasi-Yadkori et al., 2011): the batch learning model and the rare policy switch model. More specifically, in the batch learning model (Perchet et al., 2016), the agent is forced to pre-determine the number of batches (or equivalently batch size). Within each batch, the same policy is used to select actions, and the policy is updated only at the end of this batch. The amount of adaptivity in the batch learning model is measured by the number of batches, which is expected to be as small as possible. In contrast, in the rare policy switch model (Abbasi-Yadkori et al., 2011), the agent can adaptively choose when to switch the policy and therefore start a new batch in the learning process as long as the total number of policy updates does not exceed the given budget on the number of policy switches. The amount of adaptivity in the rare policy switch model can be measured by the number of policy switches, which turns out to be the same as the global switching cost introduced in Bai et al. (2019). It is worth noting that for the same amount of adaptivity[3], the rare policy switch model can be seen as a relaxation of the batch learning model since the agent in the batch learning model can only change the policy at pre-defined time steps. In our work, for each of these limited adaptivity models, we propose a variant of the LSVI-UCB algorithm (Jin et al., 2020), which can be viewed as an RL algorithm with full adaptivity in the sense that it switches the policy at a per-episode scale. Our algorithms can attain the same regret as LSVI-UCB, yet with a substantially smaller number of batches/policy switches. This enables parallel learning and improves the large-scale deployment of RL algorithms with linear function approximation.

The main contributions of this paper are summarized as follows:

[2]We say an episodic MDP is time-inhomogeneous if its reward and transition probability are different at different stages within each episode. See Definition 3.2 for details.

[3]The number of batches in the batch learning model is comparable to the number of policy switches in the rare policy switch model.

- For the *batch learning* model, we propose an LSVI-UCB-Batch algorithm for linear MDPs and show that it enjoys an $\widetilde{O}(\sqrt{d^3H^3T} + dHT/B)$ regret, where d is the dimension of the feature mapping, H is the episode length, T is the number of interactions and B is the number of batches. Our result suggests that it suffices to use only $\sqrt{T/dH}$ batches, rather than T batches, to obtain the same regret $\widetilde{O}(\sqrt{d^3H^3T})$ achieved by LSVI-UCB (Jin et al., 2020) in the fully sequential decision model. We also prove a lower bound of the regret for this model, which suggests that the required number of batches $\widetilde{O}(\sqrt{T})$ is sharp.
- For the *rare policy switch* model, we propose an LSVI-UCB-RareSwitch algorithm for linear MDPs and show that it enjoys an $\widetilde{O}(\sqrt{d^3H^3T[1+T/(dH)]^{dH/B}})$ regret, where B is the number of policy switches. Our result implies that $dH \log T$ policy switches are sufficient to obtain the same regret $\widetilde{O}(\sqrt{d^3H^3T})$ achieved by LSVI-UCB. The number of policy switches is much smaller than that[4] of the batch learning model when T is large.

Concurrent to our work, Gao et al. (2021) proposed an algorithm achieving $\widetilde{O}(\sqrt{d^3H^3T})$ regret with a $O(dH \log K)$ global switching cost in the rare policy switch model. They also proved a $\Omega(dH/\log d)$ lower bound on the global switching cost. The focus of our paper is different from theirs: our goal is to design efficient RL algorithms under a switching cost budget B, while their goal is to achieve the optimal rate in terms of T with as little switching cost as possible. On the other hand, for the rare policy switch model, our proposed algorithm (LSVI-UCB-RareSwitch) along its regret bound can imply their results by optimizing our regret bound concerning the switching cost budget B.

The rest of the paper is organized as follows. In Section 2 we discuss previous works related to this paper, with a focus on RL with linear function approximation and online learning with limited adaptivity. In Section 3 we introduce necessary preliminaries for MDPs and adaptivity constraints. Sections 4 and 5 present our proposed algorithms and the corresponding theoretical results for the batch learning model and the rare policy switch model respectively. In Section 6 we present the numerical experiment which supports our theory. Finally, we conclude our paper and point out a future direction in Section 7.

Notation We use lower case letters to denote scalars and use lower and upper case boldface letters to denote vectors and matrices respectively. For any real number a, we write $[a]^+ = \max(a, 0)$. For a vector $\mathbf{x} \in \mathbb{R}^d$ and matrix $\mathbf{\Sigma} \in \mathbb{R}^{d\times d}$, we denote by $\|\mathbf{x}\|_2$ the Euclidean norm and define $\|\mathbf{x}\|_{\mathbf{\Sigma}} = \sqrt{\mathbf{x}^\top \mathbf{\Sigma}\mathbf{x}}$. For any positive integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$. For any finite set A, we denote by $|A|$ the cardinality of A. For two sequences $\{a_n\}$ and $\{b_n\}$, we write $a_n = O(b_n)$ if there exists an absolute constant C such that $a_n \leq Cb_n$, and we write $a_n = \Omega(b_n)$ if there exists an absolute constant C such that $a_n \geq Cb_n$. We use $\widetilde{O}(\cdot)$ to further hide the logarithmic factors.

2 Related Works

Reinforcement Learning with Linear Function Approximation Recently, there have been many advances in RL with function approximation, especially the linear case. Jin et al. (2020) proposed an efficient algorithm for the first time for linear MDPs of which the transition probability and the rewards are both linear functions with respect to a feature mapping $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. Under similar assumptions, different settings (e.g., discounted MDPs) have also been studied in Yang and Wang (2019); Du et al. (2020); Zanette et al. (2020); Neu and Pike-Burke (2020) and He et al. (2021). A parallel line of work studies linear mixture MDPs (a.k.a. linear kernel MDPs) based on a ternary feature mapping $\psi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ (see Jia et al. (2020); Zhou et al. (2021b); Cai et al. (2020); Zhou et al. (2021a)). For other function approximation settings, we refer readers to generalized linear model (Wang et al., 2021), general function approximation with Eluder dimension (Wang et al., 2020; Ayoub et al., 2020), kernel approximation (Yang et al., 2020), function approximation with disagreement coefficients (Foster et al., 2021) and bilinear classes (Du et al., 2021).

Online Learning with Limited Adaptivity As we mentioned before, online learning with limited adaptivity has been studied in two popular models of adaptivity constraints: the batch learning model and the rare policy switch model.

[4]The number of policy switches is identical to the number of batches in the batch learning model.

For the *batch learning model*, Altschuler and Talwar (2018) proved that the optimal regret bound for prediction-from-experts (PFE) is $\widetilde{O}(\sqrt{T\log n})$ when the number of batches $B = \Omega(\sqrt{T\log n})$, and $\min(\widetilde{O}(T\log n/B), T)$ when $B = O(\sqrt{T\log n})$, exhibiting a phase-transition phenomenon[5]. Here T is the number of rounds and n is the number of actions. For general online convex optimization, Chen et al. (2020) showed that the minimax regret bound is $\widetilde{O}(T/\sqrt{B})$. Perchet et al. (2016) studied batched 2-arm bandits, and Gao et al. (2019) studied the batched multi-armed bandits (MAB). Dekel et al. (2014) proved a $\Omega(T/\sqrt{B})$ lower bound for batched MAB, and Altschuler and Talwar (2018) further characterized the dependence on the number of actions n and showed that the corresponding minimax regret bound is $\min(\widetilde{O}(T\sqrt{n}/\sqrt{B}), T)$. For batched linear bandits with adversarial contexts, Han et al. (2020) showed that the minimax regret bound is $\widetilde{O}(\sqrt{dT}+dT/B)$ where d is the dimension of the context vectors. Better rates can be achieved for batched linear bandits with stochastic contexts as shown in Esfandiari et al. (2021); Han et al. (2020); Ruan et al. (2020).

For the *rare policy switch model*, the minimax optimal regret bound for PFE is $O(\sqrt{T\log n})$ in terms of both the expected regret (Kalai and Vempala, 2005; Geulen et al., 2010; Cesa-Bianchi et al., 2013; Devroye et al., 2015) and high-probability guarantees (Altschuler and Talwar, 2018), where T is the number of rounds, and n is the number of possible actions. For MAB, the minimax regret bound has been shown to be $\widetilde{O}(T^{2/3}n^{1/3})$ by Arora et al. (2012); Dekel et al. (2014). For stochastic linear bandits, Abbasi-Yadkori et al. (2011) proposed a rarely switching OFUL algorithm achieving $\widetilde{O}(d\sqrt{T})$ regret with $\log(T)$ batches. Ruan et al. (2020) proposed an algorithm achieving $\widetilde{O}(\sqrt{dT})$ regret with less than $O(d\log d\log T)$ batches for stochastic linear bandits with adversarial contexts.

For episodic RL with finite state and action space, Bai et al. (2019) proposed an algorithm achieving $\widetilde{O}(\sqrt{H^3SAT})$ regret with $O(H^3SA\log(T/(AH)))$ local switching cost where S and A are the number of states and actions respectively. They also provided a $\Omega(HSA)$ lower bound on the local switching cost that is necessary for sublinear regret. For the global switching cost, Zhang et al. (2021) proposed an MVP algorithm with at most $O(SA\log(KH))$ global switching cost for time-homogeneous tabular MDPs.

3 Preliminaries

3.1 Markov Decision Processes

We consider the time-inhomogeneous episodic Markov decision process, which is denoted by a tuple $M(\mathcal{S}, \mathcal{A}, H, \{r_h\}_{h\in[H]}, \{\mathbb{P}_h\}_{h\in[H]})$. Here $\mathcal{S}$ is the state space (may be infinite), $\mathcal{A}$ is the action space where we allow the feasible action set to change from step to step, H is the length of each episode, and $r_h : \mathcal{S}\times\mathcal{A} \to [0,1]$ is the reward function for each stage $h \in [H]$. At each stage $h \in [H]$, $\mathbb{P}_h(s'|s,a)$ is the transition probability function which represents the probability for state s to transit to state s' given action a. A policy π consists of H mappings, $\{\pi_h : \mathcal{S} \to \mathcal{A}\}_{h\in[H]}$. For any policy π, we define the action-value function $Q_h^\pi(s,a)$ and value function $V_h^\pi(s)$ as follows:

$$Q_h^\pi(s,a) = r_h(s,a) + \mathbb{E}_\pi\left[\sum_{i=h}^{H} r_i(s_i,a_i)\Big| s_h = s, a_h = a\right], \qquad V_h^\pi(s) = Q_h^\pi(s,\pi_h(s)),$$

where $a_i \sim \pi_i(\cdot|s_i)$ and $s_{i+1} \sim \mathbb{P}_i(\cdot|s_i,a_i)$. The optimal value function V_h^* and the optimal action-value function $Q_h^*(s,a)$ are defined as $V^*(s) = \sup_\pi V_h^\pi(s)$ and $Q_h^*(s,a) = \sup_\pi Q_h^\pi(s,a)$, respectively. For simplicity, for any function $V : \mathcal{S} \to \mathbb{R}$, we denote $[\mathbb{P}V](s,a) = \mathbb{E}_{s'\sim\mathbb{P}(\cdot|s,a)}V(s')$. In the online learning setting, at the beginning of k-th episode, the agent chooses a policy π^k and the environment selects an initial state s_1^k, then the agent interacts with environment following policy π^k and receives states s_h^k and rewards $r_h(s_h^k, a_h^k)$ for $h \in [H]$. To measure the performance of the algorithm, we adopt the following notion of the total regret, which is the summation of suboptimalities between policy π^k and optimal policy π^*:

Definition 3.1. We denote $T = KH$, and the regret $\mathrm{Regret}(T)$ is defined as

$$\mathrm{Regret}(T) = \sum_{k=1}^{K}\left[V_1^*(s_1^k) - V_1^{\pi^k}(s_1^k)\right].$$

[5]They call it B-switching budget setting, which is identical to the batch learning model.

3.2 Linear Function Approximation

In this work, we consider a special class of MDPs called *linear MDPs* (Yang and Wang, 2019; Jin et al., 2020), where both the transition probability function and reward function can be represented as a linear function of a given feature mapping $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. Formally speaking, we have the following definition for linear MDPs.

Definition 3.2. $M(\mathcal{S}, \mathcal{A}, H, \{r_h\}_{h\in[H]}, \{\mathbb{P}_h\}_{h\in[H]})$ is called a linear MDP if there exist a *known* feature mapping $\boldsymbol{\phi}(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, *unknown* measures $\{\boldsymbol{\mu}_h = (\mu_h^{(1)}, \cdots, \mu_h^{(d)})\}_{h\in[H]}$ over $\mathcal{S}$ and unknown vectors $\{\boldsymbol{\theta}_h \in \mathbb{R}^d\}_{h\in[H]}$ with $\max_{h\in[H]}\{\|\boldsymbol{\mu}_h(\mathcal{S})\|_2, \|\boldsymbol{\theta}_h\|\} \leq \sqrt{d}$, such that the following holds for all $h \in [H]$:

- For any state-action-state triplet $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, $\mathbb{P}_h(s'|s, a) = \langle\boldsymbol{\phi}(s, a), \boldsymbol{\mu}_h(s')\rangle$.
- For any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, $r_h(s, a) = \langle\boldsymbol{\phi}(s, a), \boldsymbol{\theta}_h\rangle$.

Without loss of generality, we also assume that $\|\boldsymbol{\phi}(s, a)\|_2 \leq 1$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$.

With Definition 3.2, it is shown in Jin et al. (2020) that the action-value function can be written as a linear function of the features.

Proposition 3.3 (Proposition 2.3, Jin et al. 2020)**.** For a linear MDP, for any policy π, there exist weight vectors $\{\mathbf{w}_h^\pi\}_{h\in[H]}$ such that for any $(s, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$, we have $Q_h^\pi(s, a) = \langle\boldsymbol{\phi}(s, a), \mathbf{w}_h^\pi\rangle$. Moreover, we have $\|\mathbf{w}_h^\pi\|_2 \leq 2H\sqrt{d}$ for all $h \in [H]$.

Therefore, with the known feature mapping $\boldsymbol{\phi}(\cdot, \cdot)$, it suffices to estimate the weight vectors $\{\mathbf{w}_h^\pi\}_{h\in[H]}$ in order to recover the action-value functions. This is the core idea behind almost all the algorithms and theoretical analyses for linear MDPs.

3.3 Models for Limited Adaptivity

In this work, we consider RL algorithms with limited adaptivity. There are two typical models for online learning with such limited adaptivity: *batch learning model* (Perchet et al., 2016) and *rare policy switch model* (Abbasi-Yadkori et al., 2011).

For the batch learning model, the agent pre-determines the batch grids $1 = t_1 < t_2 < \cdots < t_B < t_{B+1} = K + 1$ at the beginning of the algorithm, where B is the number of batches. The b-th batch consists of t_b-th to $(t_{b+1} - 1)$-th episodes, and the agent follows the same policy within each batch. The adaptivity is measured by the number of batches.

For the rare policy switch model, the agent can decide whether she wants to switch the current policy or not. The adaptivity is measured by the number of policy switches, which is defined as

$$N_{\text{switch}} = \sum_{k=1}^{K-1} \mathbb{1}\{\pi^k \neq \pi^{k+1}\},$$

where $\pi^k \neq \pi^{k+1}$ means that there exists some $(h, s) \in [H] \times \mathcal{S}$ such that $\pi_h^k(s) \neq \pi_h^{k+1}(s)$. It is worth noting that N_{switch} is identical to the *global switching cost* defined in (1.1).

Given a budget on the number of batches or the number of policy switches, we aim to design RL algorithms with linear function approximation that can achieve the same regret as their full adaptivity counterpart, e.g., LSVI-UCB (Jin et al., 2020).

4 RL in the Batch Learning Model

In this section, we consider RL with linear function approximation in the batch learning model, where given the number of batches B, we need to pin down the batches before the agent starts to interact with the environment.

4.1 Algorithm and Regret Analysis

We propose LSVI-UCB-Batch algorithm as displayed in Algorithm 1, which can be regarded as a variant of the LSVI-UCB algorithm proposed in Jin et al. (2020) yet with limited adaptivity.

Algorithm 1 LSVI-UCB-Batch

Require: Number of batches B, confidence radius β, regularization parameter λ

1: Set $b \leftarrow 1$, $t_i \leftarrow (i-1) \cdot \lfloor K/B \rfloor + 1, i \in [B]$
2: **for** episode $k = 1, 2, \ldots, K$ **do**
3: Receive the initial state s_1^k
4: **if** $k = t_b$ **then**
5: $b \leftarrow b + 1$, $Q_{H+1}^k(\cdot, \cdot) \leftarrow 0$
6: **for** stage $h = H, H-1, \ldots, 1$ **do**
7: $\mathbf{\Lambda}_h^k \leftarrow \sum_{\tau=1}^{k-1} \boldsymbol{\phi}(s_h^\tau, a_h^\tau) \boldsymbol{\phi}(s_h^\tau, a_h^\tau)^\top + \lambda \mathbf{I}$
8: $\mathbf{w}_h^k \leftarrow (\mathbf{\Lambda}_h^k)^{-1} \sum_{\tau=1}^{k-1} \boldsymbol{\phi}(s_h^\tau, a_h^\tau) \cdot [r_h(s_h^\tau, a_h^\tau) + \max_{a \in \mathcal{A}} Q_{h+1}^k(s_{h+1}^\tau, a)]$
9: $\Gamma_h^k(\cdot, \cdot) \leftarrow \beta \cdot [\boldsymbol{\phi}(\cdot, \cdot)^\top (\mathbf{\Lambda}_h^k)^{-1} \boldsymbol{\phi}(\cdot, \cdot)]^{1/2}$
10: $Q_h^k(\cdot, \cdot) \leftarrow \min\{\boldsymbol{\phi}(\cdot, \cdot)^\top \mathbf{w}_h^k + \Gamma_h^k(\cdot, \cdot), H - h + 1\}^+$, $\pi_h^k(\cdot) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} Q_h^k(\cdot, a)$
11: **end for**
12: **else**
13: $Q_h^k \leftarrow Q_h^{k-1}$, $\pi_h^k \leftarrow \pi_h^{k-1}$, $\forall h \in [H]$
14: **end if**
15: **for** stage $h = 1 \ldots, H$ **do**
16: Take the action $a_h^k \leftarrow \pi_h^k(s_h^k)$, receive the reward $r_h(s_h^k, a_h^k)$ and the next state s_{h+1}^k
17: **end for**
18: **end for**

Algorithm 1 takes a series of batch grids $\{t_1, \ldots, t_{B+1}\}$ as input, where the i-th batch starts at t_i and ends at $t_{i+1} - 1$. LSVI-UCB-Batch takes the uniform batch grids as its selection of grids, i.e., $t_i = (i-1) \cdot \lfloor K/B \rfloor + 1, i \in [B]$. By Proposition 3.3, we know that for each $h \in [H]$, the optimal value function Q_h^* has the linear form $\langle \boldsymbol{\phi}(\cdot, \cdot), \mathbf{w}_h^* \rangle$. Therefore, to estimate the Q_h^*, it suffices to estimate $\mathbf{w}_h^*$. At the beginning of each batch, Algorithm 1 calculates $\mathbf{w}_h^k$ as an estimate of $\mathbf{w}_h^*$ by ridge regression (Line 8). Meanwhile, in order to measure the uncertainty of $\mathbf{w}_h^k$, Algorithm 1 sets the estimate $Q_h^k(\cdot, \cdot)$ as the summation of the linear function $\langle \boldsymbol{\phi}(\cdot, \cdot), \mathbf{w}_h^k \rangle$ and a Hoeffding-type exploration bonus term $\Gamma_h^k(\cdot, \cdot)$ (Line 10), which is calculated based on the confidence radius β. Then it sets the policy π_h^k as the greedy policy with respect to Q_h^k. Within each batch, Algorithm 1 simply keeps the policy used in the previous episode without updating (Line 13). Apparently, the number of batches of Algorithm 1 is B.

Here we would like to make a comparison between our LSVI-UCB-Batch and other related algorithms. The most related algorithm is LSVI-UCB proposed in Jin et al. (2020). The main difference between LSVI-UCB-Batch and LSVI-UCB is the introduction of batches. In detail, when $B = K$, LSVI-UCB-Batch degenerates to LSVI-UCB. Another related algorithm is the SBUCB algorithm proposed by Han et al. (2020). Both LSVI-UCB-Batch and SBUCB take uniform batch grids as the selection of batches. The difference is that SBUCB is designed for linear bandits, which is a special case of episodic MDPs with $H = 1$.

The following theorem presents the regret bound of Algorithm 1.

Theorem 4.1. There exists a constant $c > 0$ such that for any $\delta \in (0, 1)$, if we set $\lambda = 1$, $\beta = cdH\sqrt{\log(2dT/\delta)}$, then under Assumption 3.2, the total regret of Algorithm 1 is bounded by

$$\begin{aligned}\text{Regret}(T) \leq & 2H\sqrt{T \log\left(\frac{2dT}{\delta}\right)} + \frac{dHT}{2B \log 2} \log\left(\frac{T}{dH} + 1\right) \\ & + 4c\sqrt{2d^3H^3T \log\left(\frac{2dT}{\delta}\right) \log\left(\frac{T}{dH} + 1\right)}\end{aligned}$$

with probability at least $1 - \delta$.

Theorem 4.1 suggests that the total regret of Algorithm 1 is bounded by $\widetilde{O}(\sqrt{d^3H^3T} + dHT/B)$. When $B = \Omega(\sqrt{T/dH})$, the regret of Algorithm 1 is $\widetilde{O}(\sqrt{d^3H^3T})$, which is the same as that of LSVI-UCB in Jin et al. (2020). However, it is worth noting that LSVI-UCB needs K batches, while Algorithm 1 only requires $\sqrt{T/dH}$ batches, which can be much smaller than K.

Algorithm 2 LSVI-UCB-RareSwitch

Require: Policy switch parameter η, confidence radius β, regularization parameter λ

1: Initialize $\mathbf{\Lambda}_h = \mathbf{\Lambda}_h^0 = \lambda \mathbf{I}_d$ for all $h \in [H]$
2: **for** episode $k = 1, 2, \ldots, K$ **do**
3: Receive the initial state s_1^k
4: **for** stage $h = 1, 2, \cdots, H$ **do**
5: $\mathbf{\Lambda}_h^k \leftarrow \sum_{\tau=1}^{k-1} \boldsymbol{\phi}(s_h^\tau, a_h^\tau) \boldsymbol{\phi}(s_h^\tau, a_h^\tau)^\top + \lambda \mathbf{I}_d$
6: **end for**
7: **if** $\exists h \in [H], \det(\mathbf{\Lambda}_h^k) > \eta \cdot \det(\mathbf{\Lambda}_h)$ **then**
8: $Q_{H+1}^k(\cdot, \cdot) \leftarrow 0$
9: **for** step $h = H, H-1, \cdots, 1$ **do**
10: $\mathbf{\Lambda}_h \leftarrow \mathbf{\Lambda}_h^k$
11: $\mathbf{w}_h^k \leftarrow (\mathbf{\Lambda}_h^k)^{-1} \sum_{\tau=1}^{k-1} \boldsymbol{\phi}(s_h^\tau, a_h^\tau) \cdot [r_h(s_h^\tau, a_h^\tau) + \max_{a \in \mathcal{A}} Q_{h+1}^k(s_{h+1}^\tau, a)]$
12: $\Gamma_h^k(\cdot, \cdot) \leftarrow \beta \cdot [\boldsymbol{\phi}(\cdot, \cdot)^\top (\mathbf{\Lambda}_h^k)^{-1} \phi(\cdot, \cdot)]^{1/2}$
13: $Q_h^k(\cdot, \cdot) \leftarrow \min\{\phi(\cdot, \cdot)^\top \mathbf{w}_h^k + \Gamma_h^k(\cdot, \cdot), H - h + 1\}^+$, $\pi_h^k(\cdot) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} Q_h^k(\cdot, a)$
14: **end for**
15: **else**
16: $Q_h^k \leftarrow Q_h^{k-1}$, $\pi_h^k \leftarrow \pi_h^{k-1}$, $\forall h \in [H]$
17: **end if**
18: **for** stage $h = 1 \ldots, H$ **do**
19: Take the action $a_h^k \leftarrow \pi_h^k(s_h^k)$, receive the reward $r_h(s_h^k, a_h^k)$ and the next state s_{h+1}^k
20: **end for**
21: **end for**

Next, we present a lower bound to show the dependency of the total regret on the number of batches for the batch learning model.

Theorem 4.2. Suppose that $B \geq (d-1)H/2$. Then for any batch learning algorithm with B batches, there exists a linear MDP such that the regret over the first T rounds is lower bounded by

$$\text{Regret}(T) = \Omega(dH\sqrt{T} + dHT/B).$$

Theorem 4.2 suggests that in order to obtain a standard $\sqrt{T}$-regret, the number of batches B should be at least in the order of $\Omega(\sqrt{T})$, which is similar to its counterpart for batched linear bandits (Han et al., 2020).

5 RL in the Rare Policy Switch Model

In this section, we consider the rare policy switch model, where the agent can adaptively choose the batch sizes according to the information collected during the learning process.

5.1 Algorithm and Regret Analysis

We first present our second algorithm, LSVI-UCB-RareSwitch, as illustrated in Algorithm 2. Again, due to the nature of linear MDPs, we only need to estimate $\mathbf{w}_h^*$ by ridge regression, and then calculate the optimistic action-value function using the Hoeffding-type exploration bonus $\Gamma_h^k(\cdot, \cdot)$ along with the confidence radius β. Note that the size of the bonus term in Q_h^k is determined by $\mathbf{\Lambda}_h^k$. Intuitively speaking, the matrix $\mathbf{\Lambda}_h^k$ in Algorithm 2 represents how much information has been learned about the underlying MDP, and the agent only needs to switch the policy after collecting a significant amount of additional information. This is reflected by the determinant of $\mathbf{\Lambda}_h^k$, and the upper confidence bound will become tighter (shrink) as $\det(\mathbf{\Lambda}_h^k)$ increases. The determinant based criterion is similar to the idea of doubling trick, which has been used in the rarely switching OFUL algorithm for stochastic linear bandits (Abbasi-Yadkori et al., 2011), UCRL2 algorithm for tabular MDPs (Jaksch et al., 2010), and UCLK/UCLK+ for linear mixture MDPs in the discounted setting (Zhou et al., 2021b,a).

As shown in Algorithm 2, for each stage $h \in [H]$ the algorithm maintains a matrix $\mathbf{\Lambda}_h$ which is updated at each policy switch (Line 10). For every $k \in [K]$, we denote by b_k the episode from which

the policy π_k is computed. This is consistent with the one defined in Algorithm 1 in Section 4. At the start of each episode k, the algorithm computes $\{\mathbf{\Lambda}_h^k\}_{h\in[H]}$ (Line 5) and then compares them with $\{\mathbf{\Lambda}_h\}_{h\in[H]}$ using the determinant-based criterion (Line 7). The agent switches the policy if there exists some $h \in [H]$ such that $\det(\mathbf{\Lambda}_h^k)$ has increased by some pre-determined parameter $\eta > 1$, followed by policy evaluation (Lines 11-13). Otherwise, the algorithm retains the previous policy (Line 16). Here the hyperparameter η controls the frequency of policy switch, and the total number of policy switches can be bounded by a function of η.

Algorithm 2 is also a variant of LSVI-UCB proposed in Jin et al. (2020). Compared with LSVI-UCB-Batch in Algorithm 1 for the batch learning model, LSVI-UCB-RareSwitch adaptively decides when to switch the policy and can be tuned by the hyperparameter η and therefore fits into the rare policy switch model.

We present the regret bound of Algorithm 2 in the following theorem.

Theorem 5.1. There exists some constant $c > 0$ such that for any $\delta \in (0, 1)$, if we set $\lambda = 1$, $\beta = cdH\sqrt{\log(2dT/\delta)}$ and $\eta = (1 + K/d)^{dH/B}$, then the number of policy switches N_{switch} in Algorithm 2 will not exceed B. Moreover, the total regret of Algorithm 2 is bounded by

$$\text{Regret}(T) \le 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)} + 2c\sqrt{2d^3H^3T}\cdot\sqrt{\left(\frac{T}{dH}+1\right)^{\frac{dH}{B}}\log\left(\frac{T}{dH}+1\right)\log\left(\frac{2dT}{\delta}\right)} \tag{5.1}$$

with probability at least $1 - \delta$.

A few remarks are in order.

Remark 5.2. Algorithm 2 needs to update the value of each $\det(\mathbf{\Lambda}_h^k)$, and thanks to the special structure of $\mathbf{\Lambda}_h^k$, this can be done efficiently by applying the matrix determinant lemma along with the Sherman Morrison formula for efficiently updating each $(\mathbf{\Lambda}_h^k)^{-1}$. For simplicity and clarity of the presentation, we do not include these details in the pseudo-code.

Remark 5.3. By ignoring the non-dominating term, Theorem 5.1 suggests that the total regret of Algorithm 2 is bounded by $\widetilde{O}(\sqrt{d^3H^3T[1 + T/(dH)]^{dH/B}})$. Also, if we are allowed to choose B, we can choose $B = \Omega(dH\log T)$ to achieve $\widetilde{O}(\sqrt{d^3H^3T})$ regret, which is the same as that of LSVI-UCB in Jin et al. (2020). This also significantly improves upon Algorithm 1 when T is sufficiently large since previously we need $B = \Omega(\sqrt{T/dH})$. Our result exhibits a trade-off between the total regret bound and the number of policy switches, i.e., as the adaptivity budget B increases, the regret bound decreases. This will also be reflected by the numerical results later in Section 6.

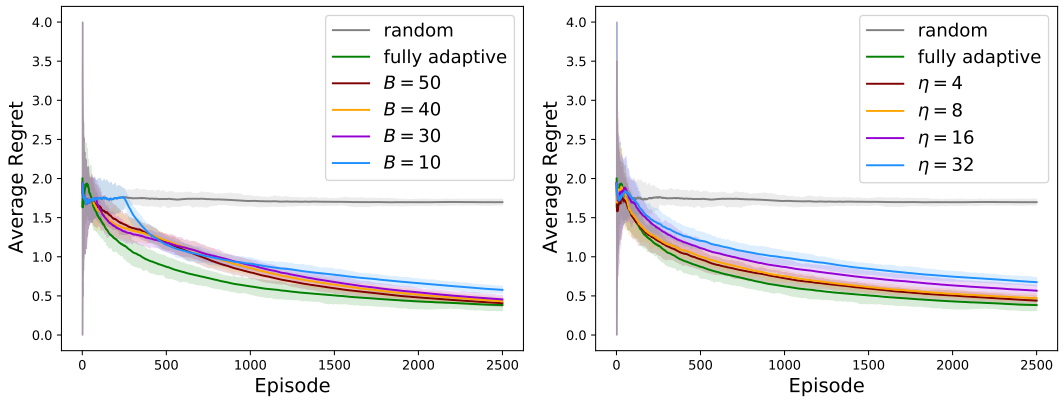
Remark 5.4. Concurrent to our work, Gao et al. (2021) proposed an algorithm with $B = \Omega(dH\log T)$ policy switches. Note that $B = \Omega(dH\log T)$ corresponds to choosing η to be a constant, which can be viewed as a special case of our algorithm. Their algorithm does not adapt to different values of budget B. Also, they did not study the batch learning model (Section 4) which we think is of equally important practical interest.

Remark 5.5. Gao et al. (2021) established a lower bound, which claims that any rare policy switch RL algorithm suffers a linear regret when $B = \widetilde{o}(dH)$. However, unlike our lower bound for the batch learning model (Theorem 4.2), their result does not provide a fine-grained regret lower bound for arbitrary adaptivity constraint B. It remains an open problem to establish such kind of lower bound for the rare policy switch model.

6 Numerical Experiment

In this section, we provide numerical experiments to support our theory. We run our algorithms, LSVI-UCB-Batch and LSVI-UCB-RareSwitch, on a synthetic linear MDP given in Example 6.1, and compare them with the fully adaptive baseline, LSVI-UCB (Jin et al., 2020).

Example 6.1 (Hard-to-learn linear MDP, Zhou et al. 2021b)**.** Let $d > 0$ be some integer and $\delta \in (0, 1)$ be a constant. The state space $\mathcal{S} = \{0, 1\}$ consists of two states, and the action space $\mathcal{A} = \{\pm 1\}^{d-3}$ contains 2^{d-3} actions where each action is represented by a $(d - 3)$-dimensional



(a) LSVI-UCB-Batch (b) LSVI-UCB-RareSwitch

Figure 1: Plot of average regret (Regret$(T)/K$) v.s. the number of episodes. The results are averaged over 50 rounds of each algorithm, and the error bars are chosen to be $[20\%, 80\%]$ empirical confidence intervals.

vector $\mathbf{a}$. For each state-action pair $(s, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$, the feature vector is given by

$$\boldsymbol{\phi}(s, a) = \begin{cases} (-\mathbf{a}^\top, 1-\delta, \delta)^\top & s = 0, \\ (0, \ldots, 0, \delta, 1-\delta) & s = 1. \end{cases} \tag{6.1}$$

For each $h \in [H]$, let $\boldsymbol{\gamma}_h \in \{\pm\delta/(d-2)\}^{d-2}$ and define the corresponding vector-valued measure as

$$\boldsymbol{\mu}_h(s) = \begin{cases} (\boldsymbol{\gamma}_h^\top, 1, 0)^\top & s = 0 \\ (-\boldsymbol{\gamma}_h^\top, 0, 1)^\top & s = 1 \end{cases}. \tag{6.2}$$

Finally, we set $\boldsymbol{\theta}_h \equiv (0, \ldots, 0, -\delta/(1-2\delta), (1-\delta)/(1-2\delta)) \in \mathbb{R}^d$ for all $h \in [H]$.

It is straightforward to verify that the feature vectors in (6.1) and the vector-valued measures in (6.2) constitute a valid linear MDP such that, for all $\mathbf{a} \in \mathcal{A}$ and $h \in [H]$,

$$r_h(s, \mathbf{a}) = \mathbb{1}\{s = 1\}, \qquad \mathbb{P}_h(s'|s, \mathbf{a}) = \begin{cases} 1 - \delta - \langle \mathbf{a}, \gamma_h \rangle & (s, s') = (0, 0), \\ \delta + \langle \mathbf{a}, \boldsymbol{\gamma}_h \rangle & (s, s') = (0, 1), \\ \delta & (s, s') = (1, 0), \\ 1 - \delta & (s, s') = (1, 1). \end{cases}$$

In our experiment[6], we set $H = 10$, $K = 2500$, $\delta = 0.35$ and $d = 13$, thus $\mathcal{A}$ contains 1024 actions. Now we apply our algorithms, LSVI-UCB-Batch and LSVI-UCB-RareSwitch, to this linear MDP instance, and compare their performance with the fully adaptive baseline LSVI-UCB (Jin et al., 2020) under different parameter settings. In detail, for LSVI-UCB-Batch, we run the algorithm for $B = 10, 20, 30, 40, 50$ respectively; for LSVI-UCB-RareSwitch, we set $\eta = 2, 4, 8, 16, 32$. We plot the average regret (Regret$(T)/K$) against the number of episodes in Figure 1. In addition to the regret of the proposed algorithms, we also plot the regret of a uniformly random policy (i.e., choosing actions uniformly randomly in each step) as a baseline.

From Figure 1, we can see that for LSVI-UCB-Batch, when $B \approx \sqrt{K}$, it achieves a similar regret as the fully adaptive LSVI-UCB as it collects more and more trajectories. For LSVI-UCB-RareSwitch, a constant value of η yields a similar order of regret compared with LSVI-UCB as suggested by Theorem 5.1. By comparing Figure 1(a) and 1(b), we can see that the performance of LSVI-UCB-RareSwitch is consistently close to that of the fully-adaptive LSVI-UCB throughout the learning process, while the performance gap between LSVI-UCB-Batch and LSVI-UCB is small only when k is large. This suggests a better adaptivity of LSVI-UCB-RareSwitch than LSVI-UCB-Batch, which only updates the policy at prefixed time steps, thus being not adaptive enough.

[6] All experiments are performed on a PC with Intel i7-9700K CPU.

Moreover, we can also see the trade-off between the regret and the adaptivity level: with more limited adaptivity (smaller B or larger η) the regret gap between our algorithms and the fully adaptive LSVI-UCB becomes larger. These results indicate that our algorithms can indeed achieve comparable performance as LSVI-UCB, even under adaptivity constraints. This corroborates our theory.

7 Conclusions

In this work, we study online RL with linear function approximation under the adaptivity constraints. We consider both the batch learning model and the rare policy switch models and propose two new algorithms LSVI-UCB-Batch and LSVI-UCB-RareSwitch for each setting. We show that LSVI-UCB-Batch enjoys an $\widetilde{O}(\sqrt{d^3H^3T} + dHT/B)$ regret and LSVI-UCB-RareSwitch enjoys an $\widetilde{O}(\sqrt{d^3H^3T[1+T/(dH)]^{dH/B}})$ regret. Compared with the fully adaptive LSVI-UCB algorithm (Jin et al., 2020), our algorithms can achieve the same regret with a much fewer number of batches/policy switches. We also prove the regret lower bound for the batch learning learning model, which suggests that the dependency on B in LSVI-UCB-Batch is tight.

For the future work, we would like to prove the regret lower bound for the rare policy switching model that explicitly depends on the given adaptivity budget B.

Acknowledgments and Disclosure of Funding

We would like to thank the anonymous reviewers for their helpful comments. Part of this work was done when DZ and QG participated the Theory of Reinforcement Learning program at the Simons Institute for the Theory of Computing in Fall 2020. DZ and QG are partially supported by the National Science Foundation CAREER Award 1906169, IIS-1904183 and AWS Machine Learning Research Award. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

ABBASI-YADKORI, Y., PÁL, D. and SZEPESVÁRI, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, vol. 24.

ALTSCHULER, J. and TALWAR, K. (2018). Online learning over a finite action set with limited switching. In *Conference On Learning Theory*. PMLR.

ARORA, R., DEKEL, O. and TEWARI, A. (2012). Online bandit learning against an adaptive adversary: from regret to policy regret. *arXiv preprint arXiv:1206.6400* .

AYOUB, A., JIA, Z., SZEPESVARI, C., WANG, M. and YANG, L. (2020). Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*. PMLR.

BAI, Y., XIE, T., JIANG, N. and WANG, Y.-X. (2019). Provably efficient q-learning with low switching cost. In *Advances in Neural Information Processing Systems*, vol. 32.

CAI, Q., YANG, Z., JIN, C. and WANG, Z. (2020). Provably efficient exploration in policy optimization. In *International Conference on Machine Learning*. PMLR.

CESA-BIANCHI, N., DEKEL, O. and SHAMIR, O. (2013). Online learning with switching costs and other adaptive adversaries. In *Advances in Neural Information Processing Systems*, vol. 26.

CHEN, L., YU, Q., LAWRENCE, H. and KARBASI, A. (2020). Minimax regret of switching-constrained online convex optimization: No phase transition. In *Advances in Neural Information Processing Systems*, vol. 33.

DEKEL, O., DING, J., KOREN, T. and PERES, Y. (2014). Bandits with switching costs: T 2/3 regret. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*.

DEVROYE, L., LUGOSI, G. and NEU, G. (2015). Random-walk perturbations for online combinatorial optimization. *IEEE Transactions on Information Theory* **61** 4099–4106.

DU, S. S., KAKADE, S. M., LEE, J. D., LOVETT, S., MAHAJAN, G., SUN, W. and WANG, R. (2021). Bilinear classes: A structural framework for provable generalization in rl. In *International Conference on Machine Learning*. PMLR.

DU, S. S., KAKADE, S. M., WANG, R. and YANG, L. F. (2020). Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*.

ESFANDIARI, H., KARBASI, A., MEHRABIAN, A. and MIRROKNI, V. (2021). Regret bounds for batched bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

FOSTER, D. J., RAKHLIN, A., SIMCHI-LEVI, D. and XU, Y. (2021). Instance-dependent complexity of contextual bandits and reinforcement learning: A disagreement-based perspective. In *Conference on Learning Theory*.

GAO, M., XIE, T., DU, S. S. and YANG, L. F. (2021). A provably efficient algorithm for linear markov decision process with low switching cost. *arXiv preprint arXiv:2101.00494* .

GAO, Z., HAN, Y., REN, Z. and ZHOU, Z. (2019). Batched multi-armed bandits problem. In *Advances in Neural Information Processing Systems*, vol. 32.

GEULEN, S., VÖCKING, B. and WINKLER, M. (2010). Regret minimization for online buffering problems using the weighted majority algorithm. In *COLT*. Citeseer.

HAN, Y., ZHOU, Z., ZHOU, Z., BLANCHET, J., GLYNN, P. W. and YE, Y. (2020). Sequential batch learning in finite-action linear contextual bandits. *arXiv preprint arXiv:2004.06321* .

HE, J., ZHOU, D. and GU, Q. (2021). Logarithmic regret for reinforcement learning with linear function approximation. In *International Conference on Machine Learning*. PMLR.

JAGHARGH, M. R. K., KRAUSE, A., LATTANZI, S. and VASSILVTISKII, S. (2019). Consistent online optimization: Convex and submodular. In *The 22nd International Conference on Artificial Intelligence and Statistics*.

JAKSCH, T., ORTNER, R. and AUER, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* **11** 1563–1600.

JIA, Z., YANG, L., SZEPESVARI, C. and WANG, M. (2020). Model-based reinforcement learning with value-targeted regression. In *Learning for Dynamics and Control*. PMLR.

JIN, C., YANG, Z., WANG, Z. and JORDAN, M. I. (2020). Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*. PMLR.

KALAI, A. and VEMPALA, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences* **71** 291–307.

LATTIMORE, T., SZEPESVARI, C. and WEISZ, G. (2020). Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*. PMLR.

NEU, G. and PIKE-BURKE, C. (2020). A unifying view of optimism in episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, vol. 33.

PERCHET, V., RIGOLLET, P., CHASSANG, S., SNOWBERG, E. ET AL. (2016). Batched bandit problems. *The Annals of Statistics* **44** 660–681.

RUAN, Y., YANG, J. and ZHOU, Y. (2020). Linear bandits with limited adaptivity and learning distributional optimal design. *arXiv preprint arXiv:2007.01980* .

WANG, R., SALAKHUTDINOV, R. R. and YANG, L. (2020). Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension. In *Advances in Neural Information Processing Systems*, vol. 33.

WANG, Y., WANG, R., DU, S. S. and KRISHNAMURTHY, A. (2021). Optimism in reinforcement learning with generalized linear function approximation. In *International Conference on Learning Representations*.

YANG, L. and WANG, M. (2019). Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*.

YANG, Z., JIN, C., WANG, Z., WANG, M. and JORDAN, M. I. (2020). On function approximation in reinforcement learning: Optimism in the face of large state spaces. In *Advances in Neural Information Processing Systems*, vol. 33.

ZANETTE, A., LAZARIC, A., KOCHENDERFER, M. and BRUNSKILL, E. (2020). Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning*. PMLR.

ZHANG, Z., JI, X. and DU, S. S. (2021). Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon. In *Conference on Learning Theory*. PMLR.

ZHOU, D., GU, Q. and SZEPESVARI, C. (2021a). Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*. PMLR.

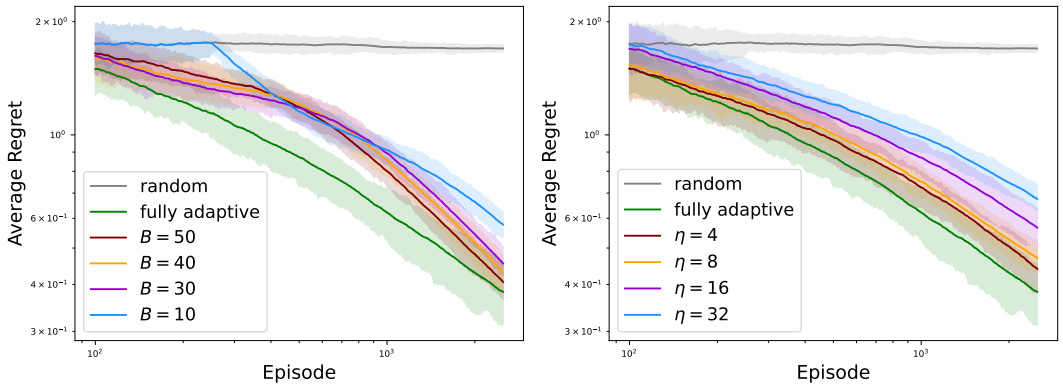
ZHOU, D., HE, J. and GU, Q. (2021b). Provably efficient reinforcement learning for discounted mdps with feature mapping. In *International Conference on Machine Learning*. PMLR.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] The goal of our paper is to develop general algorithms and theoretical analyses for RL with linear function approximation under adaptivity constraints. In this regard, we believe there are no societal impacts because this paper is mainly a theoretical work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A] We do not use any existing assets.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Our work does not involve human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]



(a) LSVI-UCB-Batch (b) LSVI-UCB-RareSwitch

Figure 2: Plot of average regret (Regret$(T)/K$) v.s. the number of episodes in log-scale. The results are averaged over 50 rounds of each algorithm, and the error bars are chosen to be $[20\%, 80\%]$ empirical confidence intervals.

A Additional Details on the Numerical Experiments

A.1 Log-scaled Plot of the Average Regret

We also provide log-scaled plot of the average regret in Figure 2. We can see that the slope of the average regret curves for our proposed algorithms is similar to that of the fully adaptive LSVI-UCB, all indicating an $\widetilde{O}(1/\sqrt{T})$ scaling.

A.2 Misspecified Linear MDP

We also empirically evaluate our algorithms on linear MDP with different levels of misspecification. In particular, based on the linear MDP instance constructed in Example 6.1, we follow the definition of ζ-approximate linear MDP in Jin et al. (2020), and consider a corrupted transition given by

$$\mathbb{P}_h(s'|0,a) = (1-f(a))\boldsymbol{\phi}(0,a)^\top \boldsymbol{\mu}_h(s') + f(a)\,\mathbb{1}\{s' = g(a)\}$$

where $f : \mathcal{A} \to [0,\zeta]$, $\zeta \in (0,1)$ and $g : \mathcal{A} \to \mathcal{S}$ are unknown. The two additional functions, f and g, can be constructed by random sampling before running the algorithms, and the magnitude of $\zeta \in (0,1)$ characterizes the level of model misspecification. All the other components of the model and the experiment configurations remain the same as those in Section 6.

Under this misspecified model with levels $\zeta = 0.05, 0.1, 0.2, 0.4$, we run LSVI-UCB-Batch with $B = 50$ and LSVI-UCB-RareSwitch with $\eta = 8$ respectively. We plot the average regret of the algorithms in Figure 3. We can see that our algorithms can still achieve a reasonably good performance under considerable levels of model misspecification.

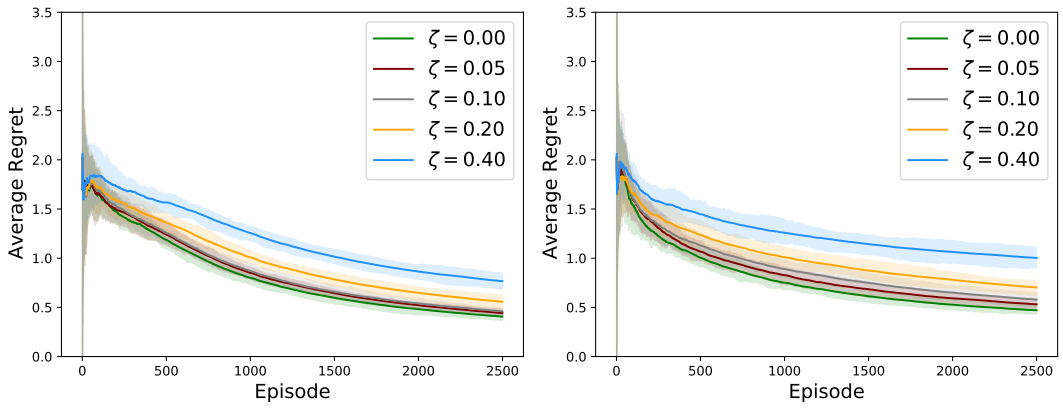
B Proofs of Theorem 4.1

In this section we prove Theorem 4.1

For simplicity, we use b_k to denote the batch t_b satisfying $t_b \le k < t_{b+1}$. Let $\Gamma_h^k(\cdot,\cdot)$ be $\beta \cdot [\boldsymbol{\phi}(\cdot,\cdot)^\top (\boldsymbol{\Lambda}_h^k)^{-1} \boldsymbol{\phi}(\cdot,\cdot)]^{1/2}$ for any $h \in [H], k \in [K]$. First, we need the following lemma which gives Regret(T) a high probability upper bound that depends on the summation of bonuses.

Lemma B.1. With probability at least $1-\delta$, the total regret of Algorithm 1 satisfies

$$\text{Regret}(T) \le \sum_{k=1}^{K}\sum_{h=1}^{H} \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\} + 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)}.$$



(a) LSVI-UCB-Batch ($B = 50$) (b) LSVI-UCB-RareSwitch ($\eta = 8$)

Figure 3: Plot of average regret ($\text{Regret}(T)/K$) v.s. the number of episodes for a misspecified linear MDP. The results are averaged over 50 rounds of each algorithm, and the error bars are chosen to be $[20\%, 80\%]$ empirical confidence intervals.

Lemma B.1 suggests that in order to bound the total regret, it suffices to bound the summation of the 'delayed' bonuses $\Gamma_h^{b_k}(s_h^k, a_h^k)$, in contrast to the per-episode bonuses $\Gamma_h^k(s_h^k, a_h^k)$ for all $k \in [K]$. The superscript b_k suggests that instead of using all the information up to the current episode k, Algorithm 1 can only use the information before the current batch b_k due to its batch learning nature. How to control the error induced by batch learning is the main difficulty in our analysis. To tackle this difficulty, we first need an upper bound for the summation of per-episode bonuses $\Gamma_h^k(s_h^k, a_h^k)$.

Lemma B.2. Let β be selected as Theorem 4.1 suggests. Then the summation of all the per-episode bonuses is bounded by

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\Gamma_h^k(s_h^k, a_h^k) \le \beta\sqrt{2dHT\log\left(\frac{T}{dH}+1\right)}.$$

It is worth noting that the per-episode bonuses are not generated from our algorithm, but instead are some virtual terms that we introduce to facilitate our analysis. Equipped with Lemma B.2, we only need to bound the difference between delayed bonuses and per-episode bonuses. We consider all the indices $(k, h) \in [K] \times [H]$. The next lemma suggests that considering the ratio between delayed bonuses and per-episode bonuses, the 'bad' indices, where the ratio is large, only appear few times. This is also the key lemma of our analysis.

Lemma B.3. Define the set $\mathcal{C}$ as follows

$$\mathcal{C} = \{(k, h) : \Gamma_h^{b_k}(s_h^k, a_h^k)/\Gamma_h^k(s_h^k, a_h^k) > 2\},$$

then we have $|\mathcal{C}| \le dHK\log(K/d+1)/(2B\log 2)$.

With all the above lemmas, we now begin to prove our main theorem.

Proof of Theorem 4.1. Suppose the event defined in Lemma B.1 holds. Then by Lemma B.1 we have that

$$\text{Regret}(T) \le \underbrace{\sum_{h=1}^{H}\sum_{k=1}^{K}\min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\}}_{I} + 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)} \tag{B.1}$$

holds with probability at least $1-\delta$. Next, we are going to bound I. Let $\mathcal{C}$ be the set defined in Lemma B.3. Then we have

$$\begin{aligned} I &= \sum_{(k,h)\in\mathcal{C}} \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\} + \sum_{(k,h)\notin\mathcal{C}} \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\} \\ &\le H|\mathcal{C}| + 4\sum_{(k,h)\notin\mathcal{C}} \Gamma_h^k(s_h^k, a_h^k) \\ &\le H|\mathcal{C}| + 4\sum_{h=1}^{H}\sum_{k=1}^{K} \Gamma_h^k(s_h^k, a_h^k), \end{aligned} \tag{B.2}$$

where the first inequality holds due to the definition of $\mathcal{C}$, and the second one holds trivially. Therefore, substituting (B.2) into (B.1), the regret can be bounded by

$$\begin{aligned} \text{Regret}(T) &\le 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)} + H|\mathcal{C}| + 4\sum_{h=1}^{H}\sum_{k=1}^{K}\Gamma_h^k(s_h^k, a_h^k) \\ &\le 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)} + \frac{dHT}{2B\log 2}\log\left(\frac{T}{dH}+1\right) \\ &\quad + 4c\sqrt{2d^3H^3T\log\left(\frac{2dT}{\delta}\right)\log\left(\frac{T}{dH}+1\right)}, \end{aligned} \tag{B.3}$$

where the second inequality holds due to Lemmas B.2 and B.3 and the fact that $T = KH$. This completes the proof. □

B.1 Proof of Lemma B.1

The following two lemmas in Jin et al. (2020) characterize the quality of the estimates given by the LSVI-UCB-type algorithms.

Lemma B.4 (Lemma B.5, Jin et al. 2020)**.** With probability at least $1-\delta$, we have $Q_h^k(s,a) \ge Q_h^*(s,a)$ for all $(s,a,h,k) \in \mathcal{S}\times\mathcal{A}\times[H]\times[K]$.

Lemma B.5 (Lemma B.4, Jin et al. 2020)**.** There exists some constant c such that if we set $\beta = cdH\sqrt{\log(dT/\delta)}$, then for any fixed policy π we have for all $(s,a,h,k)\in\mathcal{S}\times\mathcal{A}\times[H]\times[K]$ that

$$\left|\boldsymbol{\phi}(s,a)^\top(\mathbf{w}_h^{b_k} - \mathbf{w}_h^\pi) - \left(\mathbb{P}_h(V_{h+1}^{b_k} - V_{h+1}^\pi)\right)(s,a)\right| \le \beta\sqrt{\boldsymbol{\phi}(s,a)^\top(\boldsymbol{\Lambda}_h^{b_k})^{-1}\boldsymbol{\phi}(s,a)}$$

with probability at least $1-\delta$.

Proof of Lemma B.1. By Lemma B.4, we have $Q_h^k(s,a) \ge Q_h^*(s,a)$ for all $(s,a,h,k)\in\mathcal{S}\times\mathcal{A}\times[H]\times[K]$ on some event $\mathcal{E}$ such that $\mathbb{P}(\mathcal{E}) \ge 1-\delta/2$. In the following argument, all statements would be conditioned on the event $\mathcal{E}$. Then by the definition of V_1^k we know that $V_1^k(s) = \max_{a\in\mathcal{A}} Q_1^k(s,a) \ge \max_{a\in\mathcal{A}} Q_1^*(s,a) = V_1^*(s)$ for all $(s,k)\in\mathcal{S}\times[K]$. Therefore, we have

$$\text{Regret}(T) = \sum_{k=1}^{K}\left[V_1^*(s_1^k) - V_1^{\pi^k}(s_1^k)\right] \le \sum_{k=1}^{K}\left[V_1^k(s_1^k) - V_1^{\pi^k}(s_1^k)\right] = \sum_{k=1}^{K}\left[V_1^{b_k}(s_1^k) - V_1^{\pi^k}(s_1^k)\right].$$

Note that

$$V_h^{b_k}(s_h^k) - V_h^{\pi^k}(s_h^k) = Q_h^{b_k}(s_h^k, a_h^k) - Q_h^{\pi^k}(s_h^k, a_h^k),$$

which together with the definition of $Q_h^{b_k}$ and Lemma B.5 implies that

$$\begin{aligned} V_h^{b_k}(s_h^k) - V_h^{\pi^k}(s_h^k) &\le \boldsymbol{\phi}(s_h^k, a_h^k)^\top\mathbf{w}_h^{b_k} - \boldsymbol{\phi}(s_h^k, a_h^k)^\top\mathbf{w}_h^{\pi^k} + \Gamma_h^{b_k}(s_h^k, a_h^k) \\ &\le \left[\mathbb{P}_h\left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)\right](s_h^k, a_h^k) + 2\Gamma_h^{b_k}(s_h^k, a_h^k), \end{aligned}$$

where the first inequality holds due to the algorithm design, the second one holds due to Lemma B.5. Meanwhile, notice that $0 \le V_h^{b_k}(s_h^k) - V_h^*(s_h^k) \le V_h^{b_k}(s_h^k) - V_h^{\pi^k}(s_h^k) \le H$, then we have

$$\begin{aligned}
V_h^{b_k}(s_h^k) - V_h^{\pi^k}(s_h^k) &\le \min\left\{H, \left[\mathbb{P}_h\left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)\right](s_h^k, a_h^k) + 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\} \\
&\le \left[\mathbb{P}_h\left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)\right](s_h^k, a_h^k) + \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\} \\
&= V_{h+1}^{b_k}(s_{h+1}^k) - V_{h+1}^{\pi^k}(s_{h+1}^k) + \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\} \\
&\quad + \left[\mathbb{P}_h\left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)\right](s_h^k, a_h^k) - \left(V_{h+1}^{b_k}(s_{h+1}^k) - V_{h+1}^{\pi^k}(s_{h+1}^k)\right),
\end{aligned}$$

where the second inequality holds since $V_{h+1}^{b_k} - V_{h+1}^{\pi^k} \ge 0$. Recursively expand the above inequality, and we have

$$\begin{aligned}
V_1^{b_k}(s_1^k) - V_1^{\pi^k}(s_1^k) = \sum_{h=1}^H &\left\{\left[\mathbb{P}_h\left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)\right](s_h^k, a_h^k) - \left(V_{h+1}^{b_k}(s_{h+1}^k) - V_{h+1}^{\pi^k}(s_{h+1}^k)\right)\right\} \\
&+ \sum_{h=1}^H \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\}.
\end{aligned}$$

Therefore, the total regret can be bounded as follows

$$\begin{aligned}
\text{Regret}(T) \le \sum_{k=1}^K \sum_{h=1}^H &\left\{\left[\mathbb{P}_h\left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)\right](s_h^k, a_h^k) - \left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)(s_{h+1}^k)\right\} \\
&+ \sum_{k=1}^K \sum_{h=1}^H \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\}.
\end{aligned}$$

Note that conditional on $\mathcal{F}_{k,h,1}$, $V_{h+1}^{b_k}$ and $V_{h+1}^{\pi^k}$ are both deterministic, while s_{h+1}^k follows the distribution $\mathbb{P}_h(\cdot|s_h^k, a_h^k)$. Therefore, the first term on the RHS is a sum of a martingale difference sequence such that each summand has absolute value at most $2H$. Applying Azuma-Hoeffding inequaliy yields

$$\sum_{k=1}^K \sum_{h=1}^H \left\{\left[\mathbb{P}_h\left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)\right](s_h^k, a_h^k) - \left(V_{h+1}^{b_k} - V_{h+1}^{\pi^k}\right)(s_{h+1}^k)\right\} \le 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)},$$

with probability at least $1 - \delta/2$. By a union bound over the event $\mathcal{E}$ and the convergence of the martingale, with probability at least $1 - \delta$, we have

$$\text{Regret}(T) \le 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)} + \sum_{k=1}^K \sum_{h=1}^H \min\left\{H, 2\Gamma_h^{b_k}(s_h^k, a_h^k)\right\}.$$

□

B.2 Proof of Lemma B.2

We need the following lemma to bound the sum of the bonus terms.

Lemma B.6 (Lemma 11, Abbasi-Yadkori et al. 2011)**.** Let $\{\boldsymbol{\phi}_t\}_{t=1}^\infty$ be an $\mathbb{R}^d-$valued sequence. Meanwhile, let $\boldsymbol{\Lambda}_0 \in \mathbb{R}^{d\times d}$ be a positive-definite matrix and $\boldsymbol{\Lambda}_t = \boldsymbol{\Lambda}_0 + \sum_{i=1}^{t-1} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top$. It holds for any $t \in \mathbb{Z}_+$ that

$$\sum_{i=1}^t \min\{1, \boldsymbol{\phi}_i^\top \boldsymbol{\Lambda}_i^{-1} \boldsymbol{\phi}_i\} \le 2\log\left(\frac{\det(\boldsymbol{\Lambda}_{t+1})}{\det(\boldsymbol{\Lambda}_1)}\right).$$

Moreover, assuming that $\|\boldsymbol{\phi}_i\|_2 \le 1$ for all $i \in \mathbb{Z}_+$ and $\lambda_{\min}(\boldsymbol{\Lambda}_0) \ge 1$, it holds for any $t \in \mathbb{Z}_+$ that

$$\log\left(\frac{\det(\boldsymbol{\Lambda}_{t+1})}{\det(\boldsymbol{\Lambda}_1)}\right) \le \sum_{i=1}^t \boldsymbol{\phi}_i^\top \boldsymbol{\Lambda}_i^{-1} \boldsymbol{\phi}_i \le 2\log\left(\frac{\det(\boldsymbol{\Lambda}_{t+1})}{\det(\boldsymbol{\Lambda}_1)}\right).$$

Proof of Lemma B.2. We can bound the summation of $\Gamma_h^k(s_h^k, a_h^k)$ as follows:

$$\sum_{h=1}^{H}\sum_{k=1}^{K}\Gamma_h^k(s_h^k,a_h^k) \le \sum_{h=1}^{H}\sqrt{K\cdot\sum_{k=1}^{K}[\Gamma_h^k(s_h^k,a_h^k)]^2} = \beta\sqrt{K}\sum_{h=1}^{H}\sqrt{\sum_{k=1}^{K}\boldsymbol{\phi}(s_h^k,a_h^k)^\top[\boldsymbol{\Lambda}_h^k]^{-1}\boldsymbol{\phi}(s_h^k,a_h^k)},$$

where the inequality holds due to Cauchy-Schwarz inequality. Furthermore, by Lemma B.6, we have

$$\sum_{k=1}^{K}\boldsymbol{\phi}(s_h^k,a_h^k)^\top[\boldsymbol{\Lambda}_h^k]^{-1}\boldsymbol{\phi}(s_h^k,a_h^k) \le 2\log\left(\frac{\det\boldsymbol{\Lambda}_h^{K+1}}{\det\boldsymbol{\Lambda}_h^1}\right) \le 2d\log(K/d+1),$$

where the second inequality holds due to Lemma C.1. That finishes our proof. □

B.3 Proof of Lemma B.3

Proof of Lemma B.3. First, let $\mathcal{C}_h$ denote the indices k where $(k,h)\in\mathcal{C}$, then we have $|\mathcal{C}| = \sum_{h=1}^{H}|\mathcal{C}_h|$. Next we bound $|\mathcal{C}_h|$ for each h. For each $k\in\mathcal{C}_h$, suppose $t_b \le k < t_{b+1}$, then we have $b_k = t_b$ and

$$\log\det(\boldsymbol{\Lambda}_h^{t_{b+1}}) - \log\det(\boldsymbol{\Lambda}_h^{t_b}) \ge \log\det(\boldsymbol{\Lambda}_h^{k}) - \log\det(\boldsymbol{\Lambda}_h^{b_k}) \ge 2\log(\Gamma_h^{b_k}(s_h^k,a_h^k)/\Gamma_h^k(s_h^k,a_h^k)) > 2\log 2,$$

where the first inequality holds since $\boldsymbol{\Lambda}_h^{t_{b+1}} \succeq \boldsymbol{\Lambda}_h^k$, the second inequality holds due to Lemma C.2, the third one holds due to the definition of $\mathcal{C}_h$. Thus, let $\widehat{\mathcal{C}}_h$ denote the set

$$\widehat{\mathcal{C}}_h = \{b\in[B] : \log\det(\boldsymbol{\Lambda}_h^{t_{b+1}}) - \log\det(\boldsymbol{\Lambda}_h^{t_b}) > 2\log 2\},$$

we have $|\mathcal{C}_h| \le \lfloor K/B\rfloor\cdot|\widehat{\mathcal{C}}_h|$. In the following we bound $|\widehat{\mathcal{C}}_h|$. Now we consider the sequence $\{\log\det(\boldsymbol{\Lambda}_h^{t_{b+1}}) - \log\det(\boldsymbol{\Lambda}_h^{t_b})\}$. It is easy to see $\log\det(\boldsymbol{\Lambda}_h^{t_{b+1}}) - \log\det(\boldsymbol{\Lambda}_h^{t_b}) \ge 0$, therefore

$$2\log 2|\widehat{\mathcal{C}}_h| \le \sum_{b\in\widehat{\mathcal{C}}_h}[\log\det(\boldsymbol{\Lambda}_h^{t_{b+1}}) - \log\det(\boldsymbol{\Lambda}_h^{t_b})] \le \sum_{b=1}^{B}[\log\det(\boldsymbol{\Lambda}_h^{t_{b+1}}) - \log\det(\boldsymbol{\Lambda}_h^{t_b})]. \quad \text{(B.4)}$$

Meanwhile, we have

$$\sum_{b=1}^{B}[\log\det(\boldsymbol{\Lambda}_h^{t_{b+1}}) - \log\det(\boldsymbol{\Lambda}_h^{t_b})] = \log\det(\boldsymbol{\Lambda}_h^{t_{B+1}}) = \log\det(\boldsymbol{\Lambda}_h^{K+1}) \le d\log(K/d+1), \quad \text{(B.5)}$$

where the last inequality holds due to Lemma C.1. Therefore, (B.4) and (B.5) suggest that $|\widehat{\mathcal{C}}_h| \le d\log(K/d+1)/(2\log 2)$. Finally, we bound $|\mathcal{C}|$ as follows, which ends our proof.

$$|\mathcal{C}| = \sum_{h=1}^{H}|\mathcal{C}_h| \le \sum_{h=1}^{H}K/B\cdot|\widehat{\mathcal{C}}_h| \le dHK\log(K/d+1)/(2B\log 2).$$

□

C Proof of Theorem 5.1

Now we provide the proof of Theorem 5.1. We continue to use the notions that have been introduced in Section 4. We first give an upper bound on the determinant of $\boldsymbol{\Lambda}_h^k$.

Lemma C.1. Let $\{\boldsymbol{\Lambda}_h^k, (k,h)\in[K]\times[H]\}$ be as defined in Algorithms 1 and 2. Then for all $h\in[H]$ and $k\in[K]$, we have $\det(\boldsymbol{\Lambda}_h^k) \le (\lambda + (k-1)/d)^d$.

Proof. Note that

$$\mathrm{tr}(\boldsymbol{\Lambda}_h^k) = \mathrm{tr}(\lambda\mathbf{I}_d) + \sum_{\tau=1}^{k-1}\mathrm{tr}\left(\boldsymbol{\phi}(s_h^\tau,a_h^\tau)\boldsymbol{\phi}(s_h^\tau,a_h^\tau)^\top\right) = \lambda d + \sum_{\tau=1}^{k-1}\|\boldsymbol{\phi}(s_h^\tau,a_h^\tau)\|_2^2 \le \lambda d + k - 1,$$

where the inequality follows from the assumption that $\|\boldsymbol{\phi}(s,a)\|_2 \leq 1$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Since $\boldsymbol{\Lambda}_h^k$ is positive semi-definite, by inequality of arithmetic and geometric means, we have

$$\det(\boldsymbol{\Lambda}_h^k) \leq \left(\frac{\mathrm{tr}(\boldsymbol{\Lambda}_h^k)}{d}\right)^d \leq \left(\lambda + \frac{k-1}{d}\right)^d.$$

This finishes the proof. □

Next lemma provides a determinant-based upper bound for the ratio between the norms $\|\cdot\|_{\mathbf{A}}$ and $\|\cdot\|_{\mathbf{B}}$, where $\mathbf{A} \succeq \mathbf{B}$.

Lemma C.2 (Lemma 12, Abbasi-Yadkori et al. 2011)**.** Suppose $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d\times d}$ are two positive definite matrices satisfying that $\mathbf{A} \succeq \mathbf{B}$, then for any $\mathbf{x} \in \mathbb{R}^d$, we have $\|\mathbf{x}\|_{\mathbf{A}} \leq \|\mathbf{x}\|_{\mathbf{B}} \cdot \sqrt{\det(\mathbf{A})/\det(\mathbf{B})}$.

The switching cost of Algorithm 2 is characterized in the following lemma.

Lemma C.3. For any $\eta > 1$ and $\lambda > 0$, the global switching cost of Algorithm 2 is bounded by

$$N_{\text{switch}} \leq \frac{dH}{\log \eta} \log\left(1 + \frac{K}{\lambda d}\right).$$

Proof. Let $\{k_1, k_2, \cdots, k_{N_{\text{switch}}}\}$ be the episodes where the algorithm updates the policy, and we also define $k_0 = 0$. Then by the determinant-based criterion (Line 7), for each $i \in [N_{\text{switch}}]$ there exists at least one $h \in [H]$ such that

$$\det(\boldsymbol{\Lambda}_h^{k_i}) > \eta \cdot \det(\boldsymbol{\Lambda}_h^{k_{i-1}}).$$

By the definition of $\boldsymbol{\Lambda}_h^k$ (Line 5), we know that $\boldsymbol{\Lambda}_h^{j_1} \succeq \boldsymbol{\Lambda}_h^{j_2}$ for all $j_1 \geq j_2$ and $h \in [H]$. Thus we further have

$$\prod_{h=1}^H \det(\boldsymbol{\Lambda}_h^{k_i}) > \eta \cdot \prod_{h=1}^H \det(\boldsymbol{\Lambda}_h^{k_{i-1}}).$$

Applying the above inequality for all $i \in [N_{\text{switch}}]$ yields

$$\prod_{h=1}^H \det\left(\boldsymbol{\Lambda}_h^{k_{N_{\text{switch}}}}\right) > \eta^{N_{\text{switch}}} \cdot \prod_{h=1}^H \det(\boldsymbol{\Lambda}_h^0) = \eta^{N_{\text{switch}}} \lambda^{dH},$$

as we initialize $\boldsymbol{\Lambda}_h^0$ to be $\lambda \mathbf{I}_d$. While by Lemma C.1, we have

$$\prod_{h=1}^H \det\left(\boldsymbol{\Lambda}_h^{k_{N_{\text{switch}}}}\right) \leq \prod_{h=1}^H \det(\boldsymbol{\Lambda}_h^K) \leq \left(\lambda + \frac{K}{d}\right)^{dH}.$$

Therefore, combining the above two inequalities, we obtain that

$$N_{\text{switch}} \leq \frac{dH}{\log \eta} \log\left(1 + \frac{K}{\lambda d}\right).$$

This completes the proof. □

We now begin to prove our main theorem.

Proof of Theorem 5.1. First, substituting the choice of η and $\lambda = 1$ into the bound in Lemma C.3 yields that $N_{\text{switch}} \leq B$.

Next, we bound the regret of Algorithm 2. The result of Lemma B.1 still holds here, thus it suffices to bound the summation of the bonus terms $\Gamma_h^{b_k}(s_h^k, a_h^k)$. Note that $b_k \leq k$, and thus $\boldsymbol{\Lambda}_h^k \succeq \boldsymbol{\Lambda}_h^{b_k}$ for all $(h,k) \in [H] \times [K]$. Then by Lemma C.2 we have

$$\frac{\Gamma_h^{b_k}(s_h^k, a_h^k)}{\Gamma_h^k(s_h^k, a_h^k)} \leq \sqrt{\frac{\det(\boldsymbol{\Lambda}_h^k)}{\det(\boldsymbol{\Lambda}_h^{b_k})}} \leq \sqrt{\eta} \tag{C.1}$$

for all $(h,k) \in [H] \times [K]$, where the second inequality holds due to the algorithm design. Hence, we have

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\Gamma_h^{b_k}(s_h^k,a_h^k) \le \sqrt{\eta}\sum_{k=1}^{K}\sum_{h=1}^{H}\Gamma_h^k(s_h^k,a_h^k) \le \beta\sqrt{2\eta dHT\log\left(\frac{T}{dH}+1\right)},$$

where the second inequality follows from Lemma B.2. Therefore, we conclude by Lemma B.1 that

$$\text{Regret}(T) \le 2c\sqrt{2\eta d^3H^3T\log\left(\frac{T}{dH}+1\right)\log\left(\frac{2dT}{\delta}\right)} + 2H\sqrt{T\log\left(\frac{2dT}{\delta}\right)} \tag{C.2}$$

holds with probability at least $1-\delta$. Finally, substituting the choice of η into (C.2) finishes our proof. □

D Proofs of Theorem 4.2

In this section, we prove the lower bound for the batch learning model.

Proof of Theorem 4.2. We prove the $\Omega(dH\sqrt{T})$ and $\Omega(dHT/B)$ lower bounds separately. The first term has been proved in Theorem 5.6, (Zhou et al., 2021a). In the remaining of this proof, we prove the second term. We consider a class of MDPs parameterized by $\boldsymbol{\gamma} \in \Gamma \subset \mathbb{R}^{2dH}$, where Γ is defined as follows

$$\Gamma = \big\{(\mathbf{b}_{1,1}^\top,\cdots,\mathbf{b}_{H,d}^\top)^\top : \mathbf{b}_{i,j} \in \{(0,1)^\top,(1,0)^\top\}\big\}.$$

The MDP is defined as follows. The states space $\mathcal{S}$ consist of has $d+1$ states $x_0,\cdots,x_d$, and the action space $\mathcal{A}$ contains two actions $\mathbf{a}_1 = (0,1)^\top, \mathbf{a}_2 = (1,0)^\top$. For any $\boldsymbol{\gamma} = (\mathbf{b}_{1,1}^\top,\cdots,\mathbf{b}_{H,d}^\top)^\top$, the feature mapping is defined as

$$\boldsymbol{\phi}(x_0,\mathbf{a}_j) = (1,\underbrace{0,\cdots,0}_{2d})^\top, \qquad \boldsymbol{\phi}(x_i,\mathbf{a}_j) = (1,\underbrace{0,\cdots,0}_{2i-2},\mathbf{a}_j^\top,\underbrace{0,\cdots,0}_{2d-2i})^\top \in \mathbb{R}^{2d+1}$$

for every $i \in [d]$ and $j \in \{1,2\}$. We further define the vector-valued measures as

$$\boldsymbol{\mu}_h^{\boldsymbol{\gamma}}(x_0) = (1,-\mathbf{b}_{h,1}^\top,\cdots,-\mathbf{b}_{h,d}^\top)^\top, \qquad \boldsymbol{\mu}_h^{\boldsymbol{\gamma}}(x_i) = (\underbrace{0,\cdots,0}_{2i-1},\mathbf{b}_{h,i}^\top,\underbrace{0,\cdots,0}_{2d-2i})^\top$$

for every $i \in [d]$, $j \in \{1,2\}$ and $h \in [H]$. Finally, for each $h \in [H]$, we define

$$\boldsymbol{\theta}_h = (0,\underbrace{1,\cdots,1}_{2d})^\top \in \mathbb{R}^{2d+1}.$$

Thereby, for each $h \in [H]$, the transition $\mathbb{P}_h^{\boldsymbol{\gamma}}$ is defined as $\mathbb{P}_h^{\boldsymbol{\gamma}}(s'|s,\mathbf{a}) = \langle\boldsymbol{\phi}(s,\mathbf{a}),\boldsymbol{\mu}_h^{\boldsymbol{\gamma}}(s')\rangle$, and the reward function is $r_h(s,\mathbf{a}) = \langle\boldsymbol{\phi}(s,\mathbf{a}),\boldsymbol{\theta}_h\rangle$ for all $(s,\mathbf{a}) \in \mathcal{S}\times\mathcal{A}$. It is straightforward to see that the reward satisfies $r_h(x_0,\mathbf{a}) = 0$ and $r_h(x_i,\mathbf{a}) = 1$ for $i \in [d]$ and all $\mathbf{a} \in \mathcal{A}$. In addition, the starting state can be x_0 or x_i.

Based on the above definition, we have the following transition dynamic:

- x_0 is an absorbing state.
- For any $i \in [d]$, x_i can only transit to x_0 or x_i.
- For any episode starting from x_0, there is no regret.
- For any episode starting from some x_i with $i \in [d]$, suppose h is the first stage where the agent did not choose the "right" action $\mathbf{a} = \mathbf{b}_{h,i}$, then the regret for this episode is $H-h$.

Now we show that for any deterministic algorithm[7], there exists a $\boldsymbol{\gamma} \in \Gamma$ such that the regret is lower bounded by dHT/B. Suppose $1 = t_1 < \cdots < t_{B+1} = K+1$. We can treat all episodes in the same

[7]The lower bound of random algorithms is lower bounded by the lower bound of deterministic algorithms according to Yao's minimax principle.

batch as copies of one episode, because all actions taken by the agent, transitions and rewards are the same. When $B \geq dH$, there exists $\mathcal{C} = \{c_{1,1}, \cdots, c_{H,d}\} \subset [B]$ with $|\mathcal{C}| = dH$ such that

$$\sum_{h\in[H]} \sum_{j\in[d]} (t_{c_{h,j}+1} - t_{c_{h,j}}) \geq \frac{dHK}{B}.$$

For simplicity, we denote the i-th batch as the collection of episodes $\{t_i, \cdots, t_{i+1} - 1\}$. Now we carefully pick the starting state s_0^i for the episodes in the i-th batch.

- For any batch whose starting episode does not belong to $\mathcal{C}$, we set the starting states of the episodes in this batch as x_0. In other words, for $i \notin \mathcal{C}$, we set $s_0^{t_i} = \cdots = s_0^{t_{i+1}-1} = x_0$.
- For any batch whose starting episode lies in $\mathcal{C}$, for $i = c_{h,j} \in \mathcal{C}$, we set $s_0^{t_{c_{h,j}}} = \cdots = s_0^{t_{c_{h,j}+1}-1} = x_j$.

We consider the regret over batches $c_{1,i}, \cdots, c_{H,i}$. Since the algorithm, transition and reward are all deterministic, then the environment can predict the agent's selection. Specifically, suppose the agent will always take action $\mathbf{a}$ at h-th stage in the episodes belonging to the $c_{h,j}$-th batch, where $h \leq H/2$. Then the environment selects $\mathbf{b}_{h,j}$ as $(1,1)^\top - \mathbf{a}$, i.e., the other action. Therefore, the agent will always pick the "wrong" action when she firstly visits state x_j at h-th stage, which occurs at least $H - h \geq H/2$ regret. Moreover, since for the batch learning model, all the actions are decided at the beginning of each batch, then the $H/2$ regret will last $(t_{c_{h,j}+1} - t_{c_{h,j}})$ episodes. Taking the summation, we have

$$\text{Regret}(T) \geq \frac{H}{2} \cdot \sum_{h\in[H]} \sum_{j\in[d]} (t_{c_{h,j}+1} - t_{c_{h,j}}) \geq \frac{dHT}{2B}.$$

Finally, replacing d by $(d-1)/2$, we can convert our feature mapping from a $(2d+1)$-dimensional vector to a d-dimensional vector and complete the proof. □