Efficient Local Planning with Linear Function Approximation

Dong Yin DONGYIN@GOOGLE.COM
Botao Hao BHAO@GOOGLE.COM
Yasin Abbasi-Yadkori YADKORI@GOOGLE.COM
Nevena Lazić NEVENA@GOOGLE.COM
DeepMind

Csaba Szepesvári SZEPI@GOOGLE.COM
DeepMind and University of Alberta

Editors: Sanjoy Dasgupta and Nika Haghtalab

Abstract

We study query and computationally efficient planning algorithms for discounted Markov decision processes (MDPs) with linear function approximation and a simulator. The agent is assumed to have *local access* to the simulator, meaning that the simulator can be queried only at states that have been encountered in previous steps. We propose two new algorithms for this setting, which we call *confident Monte Carlo least-squares policy iteration* (CONFIDENT MC-LSPI), and *confident Monte Carlo Politex* (CONFIDENT MC-POLITEX), respectively. The main novelty in our algorithms is that they gradually build a set of state-action pairs ("core set") with which it can control the extrapolation errors. We show that our algorithms have polynomial query and computational cost in the dimension of the features, the effective planning horizon and the targeted sub-optimality, while the cost remains independent of the size of the state space. An interesting technical contribution of our work is the introduction of a novel proof technique that makes use of a *virtual policy iteration* algorithm. We use this method to leverage existing results on approximate policy iteration with ℓ_∞-bounded error to show that our algorithm can learn the optimal policy for the given initial state even only with local access to the simulator. We believe that this technique can be extended to broader settings beyond this work.

Keywords: local access, planning, linear function approximation, policy iteration

1. Introduction

Efficient planning lies at the heart of modern reinforcement learning (RL). In the simulation-based RL, the agent has access to a simulator which it uses to query a state-action pair to obtain the reward of the queried pair and the next state. When planning with large state spaces in the presence of features, the agent can also compute the feature vector associated with a state or a state-action pair. Planning efficiency is measured in two ways: using *query cost*, the number of calls to the simulator, and using *computation cost*, the total number of logical and arithmetic operations that the agent uses. In Markov decision processes (MDPs) with a large state space, we call a planning algorithm *query-efficient* (*computationally-efficient*) if its query (respectively, computational) cost is independent of the size of the state space and polynomial in other parameters of the problem such as the dimension of the feature space, the effective planning horizon, the number of actions and the targeted sub-optimality.

Prior works on planning in MDPs often assume that the agent has access to a *generative model* which allows the agent to query the simulator with any *arbitrary* state-action pair (Kakade, 2003; Sidford et al., 2018; Yang and Wang, 2019; Lattimore et al., 2020). In what follows, we will call

this the *random access* model. The random access model is often difficult to support. To illustrate this, consider a problem where the goal is to move the joints of a robot arm so that it moves objects around. The simulation state in this scenario is then completely described by the position, orientation and associated velocities of the various rigid objects involved. To access a state, a planner can then try to choose some values for each of the variables involved. Unfortunately, given only box constraints on the variable values (as is typically the case), a generic planner will often choose value combinations that are invalid based on physics, for example with objects penetrating each other in space. This problem is not specific to robotic applications but also arises in MDPs corresponding to combinatorial search, just to mention a second example.

To address this challenge, we replace the random access model with a *local access* model, where the only states at which the agent can query the simulator are the initial states provided to the agent, or states returned in response to previously issued queries. This access model can be implemented with any simulator that supports resetting its internal state to a previously stored such state. This type of checkpointing is widely supported, and if a simulator does not support it, there are general techniques that can be applied to achieve this functionality. As such, this access model significantly expands the scope of planners.

Definition 1 (Local access to the simulator) *We say the agent has local access to the simulator if the agent is allowed to query the simulator with a state that the agent has previously seen paired with an arbitrary action.*

Our work relies on linear function approximation. Very recently, Weisz et al. (2021b) showed that linear realizability assumption of the optimal state-action value function (Q^*-realizability) alone is not sufficient to develop a query-efficient planner. In this paper, we assume linear realizability of all policies (Q_π-realizability). We discuss several drawbacks of previous works (Lattimore et al., 2020; Du et al., 2020a) under the same realizability assumption. First, these works require the knowledge of the features of all state-action pairs; otherwise, the agent has to spend $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$ query cost to extract the features of all possible state-action pairs, where $|\mathcal{S}|$ and $|\mathcal{A}|$ are the sizes of the state space and action space, respectively. Second, these algorithms require the computation of either an approximation of the global optimal design (Lattimore et al., 2020) or a barycentric spanner (Du et al., 2020a) of the matrix of all features. Although there exists algorithms to approximate the optimal design (Todd, 2016) or barycentric spanner (Awerbuch and Kleinberg, 2008), the computational complexities for these algorithms are polynomial in the total number of all possible feature vectors, i.e., $|\mathcal{S}||\mathcal{A}|$, which is impractical for large MDPs.

We summarize our contributions as follows:

- With local access to the simulator, we propose two policy optimization algorithms—*confident Monte Carlo least-squares policy iteration* (CONFIDENT MC-LSPI), and its regularized (see e.g. Even-Dar et al. (2009); Abbasi-Yadkori et al. (2019)) version *confident Monte Carlo Politex* (CONFIDENT MC-POLITEX). Both of our algorithms maintain a *core set* of state-action pairs and run Monte Carlo rollouts from these pairs using the simulator. The algorithms then use the rollout results to estimate the Q-function values and then apply policy improvement. During each rollout procedure, whenever the algorithm observes a state-action pair that it is less confident about (with large uncertainty), the algorithm adds this pair to the core set and restarts. Compared to several prior works that use additive bonus (Jin et al., 2020; Cai et al., 2020), our algorithm design demonstrates that in the local access setting, core-set-based exploration is an effective approach.

- Under the Q_π-realizability assumption, we prove that both CONFIDENT MC-LSPI and CONFIDENT MC-POLITEX can learn a κ-optimal policy with query cost of $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}), \log(b))$ and computational costs of $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, |\mathcal{A}|, \log(\frac{1}{\delta}), \log(b))$, where d is the dimension of the feature of state-action pairs, γ is the discount factor, δ is the error probability, and b is the bound on the ℓ_2 norm of the linear coefficients for the Q-functions. In the presence of a model misspecification error ϵ, we show that CONFIDENT MC-LSPI achieves a final sub-optimality of $\tilde{\mathcal{O}}(\frac{\epsilon\sqrt{d}}{(1-\gamma)^2})$, whereas CONFIDENT MC-POLITEX can improve the sub-optimality to $\tilde{\mathcal{O}}(\frac{\epsilon\sqrt{d}}{1-\gamma})$ with a higher query cost.
- We develop a novel proof technique that makes use of a *virtual policy iteration* algorithm. We use this method to leverage existing results on approximate policy iteration which assumes that in each iteration, the approximation of the Q-function has a bounded ℓ_∞ error (Munos, 2003; Farahmand et al., 2010) (see Section 5 for details).

2. Related work

Simulators or generative models have been considered in early studies of reinforcement learning (Kearns and Singh, 1999; Kakade, 2003). Recently, it has been shown empirically that in the local access setting, core-set-based exploration has strong performance in hard-exploration problems (Ecoffet et al., 2019). In this section, we mostly focus on related theoretical works. We distinguish among *random access*, *local access*, and *online access*.

- Random access means that the agent is given a list of all possible state action pairs and can query any of them to get the reward and a sample of the next state.
- Local access means that the agent can access previously encountered states, which can be implemented with checkpointing. The local access model that we consider in this paper is a more practical version of planning with a simulator.
- Online access means that the simulation state can only be reset to the initial state (or distribution) or moved to a next random state given an action. The online access setting is more restrictive compared to local access, since the agent can only follow the MDP dynamics during the learning process.

We also distinguish between *offline* and *online* planning. In the offline planning problem, the agent only has access to the simulator during the training phase, and once the training is finished, the agent outputs a policy and executes the policy in the environment without access to a simulator. This is the setting that we consider in this paper. On the other hand, in the online planning problem, the agent can use the simulator during both the training and inference phases, meaning that the agent can use the simulator to choose the action when executing the policy. Usually, online RL algorithms with sublinear regret can be converted to an offline planning algorithm under the online access model with standard online-to-batch conversion (Cesa-Bianchi et al., 2004). While most of the prior works that we discuss in this section are for the offline planning problem, the TENSORPLAN algorithm (Weisz et al., 2021a) considers online planning.

In terms of notation, some works considers finite-horizon MDPs, in which case we use H to denote the episode length (similar to the effective planning horizon $(1-\gamma)^{-1}$ in infinite-horizon

discounted MDPs). Our discussion mainly focuses on the results with linear function approximation. We summarize some of the recent advances on efficient planning in large MDPs in Table 1.

Table 1: Recent advances on RL algorithms with linear function approximation under different assumptions. Positive results mean query cost depends only polynomially on the relative parameter while negative results refer an exponential lower bound on the query complexity. CE stands for computational efficiency and "no" for CE means no computational efficient algorithm is provided.
†: The algorithms in these works are not query or computationally efficient unless the agent is provided with an approximate optimal design (Lattimore et al., 2020) or barycentric spanner (Du et al., 2020a) or "core states" (Shariff and Szepesvári, 2020) for free.
‡: Weisz et al. (2021a) consider the online planning problem whereas other works in this table consider (or can be converted to) the offline planning problem.

Positive Results	Assumption	CE	Access Model
Yang and Wang (2019)	linear MDP	yes	random access
Lattimore et al. (2020); Du et al. (2020a)	Q_π-realizability	no †	random access
Shariff and Szepesvári (2020)	V^*-realizability	no †	random access
This work	**Q_π-realizability**	**yes**	**local access**
Weisz et al. (2021a) ‡	V^*-realizability, $\mathcal{O}(1)$ actions	no	local access
Li et al. (2021)	Q^*-realizability, constant gap	yes	local access
Jiang et al. (2017)	low Bellman rank	no	online access
Zanette et al. (2020)	low inherent Bellman error	no	online access
Du et al. (2021)	bilinear class	no	online access
Lazic et al. (2021); Wei et al. (2021)	Q_π-realizability, feature excitation	yes	online access
Jin et al. (2020); Agarwal et al. (2020a)	linear MDP	yes	online access
Zhou et al. (2020); Cai et al. (2020)	linear mixture MDP	?	online access
Negative Results	**Assumption**	**CE**	**Access Model**
Du et al. (2020a)	Q_π-realizability, $\epsilon = \Omega(\sqrt{H/d})$	N/A	random access
Weisz et al. (2021b)	Q^*-realizability, $\exp(d)$ actions	N/A	random access
Wang et al. (2021)	Q^*-realizability, constant gap	N/A	online access

Random access Theoretical guarantees for the random access model have been obtained for the tabular setting (Sidford et al., 2018; Agarwal et al., 2020b; Li et al., 2020; Azar et al., 2013). As for linear function approximation, different assumptions have been made for theoretical analysis. Under the linear MDP assumption, Yang and Wang (2019) derived an optimal $\mathcal{O}(d\kappa^{-2}(1-\gamma)^{-3})$ query complexity bound by a variance-reduction Q-learning type algorithm. Under the Q_π-realizability of all determinstic policies (a strictly weaker assumption than linear MDP (Zanette et al., 2020)), Du et al. (2020a) showed a negative result for the settings with model misspecification error $\epsilon = \Omega(\sqrt{H/d})$ (see also Van Roy and Dong (2019); Lattimore et al. (2020)). When $\epsilon = o((1-\gamma)^2/\sqrt{d})$, assuming the access to the full feature matrix, Lattimore et al. (2020) proposed algorithms with polynomial query costs, and Du et al. (2020a) proposed similar algorithm for the exact Q_π realizability setting. Since these works need to find a globally optimal design or barycentric spanner, their

computational costs depend polynomially on the size of the state space. Under the V^*-realizability assumption (i.e., the optimal value function is linear in some feature map), Shariff and Szepesvári (2020) proposed a planning algorithm assuming the availability of a set of core states but obtaining such core states can still be computationally inefficient. Zanette et al. (2019) proposed an algorithm that uses a similar concept named anchor points but only provided a greedy heuristic to generate these points. A notable negative result is established in Weisz et al. (2021b) that shows that with only Q^*-realizability, any agent requires $\min(\exp(\Omega(d)), \exp(\Omega(H)))$ queries to learn an optimal policy.

Local access Many prior studies have used simulators in tree-search style algorithms (Kearns et al., 2002; Munos, 2014). Under this setting, for the online planning problem, recently Weisz et al. (2021a) established an $\mathcal{O}((dH/\kappa)^{|\mathcal{A}|})$ query cost bound to learn an κ-optimal policy by the TENSORPLAN algorithm assuming the V^*-realizability. Whenever the action set is small, TENSORPLAN is query efficient, but its computational efficiency is left as an open problem. Under Q^*-realizability and constant sub-optimality gap, for the offline planning problem, Li et al. (2021) proposed an algorithm with $\mathrm{poly}(d, H, \kappa^{-1}, \Delta_{\text{gap}}^{-1})$ query and computational costs.

Online access As mentioned, many online RL algorithms can be converted to a policy optimization algorithm under the online access model using online-to-batch conversion. There is a large body of literature on online RL with linear function approximation and here we discuss a non-exhaustive list of prior works. Under the Q^*-realizability assumption, assuming that the probability transition of the MDP is deterministic, Wen and Van Roy (2013) proposed a sample and computationally efficient algorithm via the eluder dimension (Russo and Van Roy, 2013). Assuming the MDP has low Bellman rank, Jiang et al. (2017) proposed an algorithm that is sample efficient but computationally inefficient, and similar issues arise in Zanette et al. (2020) under the low inherent Bellman error assumption. Du et al. (2021) proposed a more general MDP class named *bilinear class* and provided a sample efficient algorithm, but the computational efficiency is unclear.

Under Q_π-realizability, several algorithms, such as POLITEX (Abbasi-Yadkori et al., 2019; Lazic et al., 2021), AAPI (Hao et al., 2021), and MDP-EXP2 (Wei et al., 2021) achieved sublinear regret in the infinite horizon average reward setting and are also computationally efficient. However, the corresponding analysis avoids the exploration issue by imposing a *feature excitation* assumption which may not be satisfied in many problems. Under the linear MDP assumption, Jin et al. (2020) established a $\mathcal{O}(\sqrt{d^3H^3T})$ regret bound for an optimistic least-square value iteration algorithm. Agarwal et al. (2020a) derived a $\mathrm{poly}(d, H, \kappa^{-1})$ sample cost bound for the policy cover-policy gradient algorithm, which can also be applied in the state aggregation setting; the algorithm and sample cost were subsequently improved in Zanette et al. (2021). Under the linear mixture MDP assumption (Yang and Wang, 2020; Zhou et al., 2020), Cai et al. (2020) proved an $\mathcal{O}(\sqrt{d^3H^3T})$ regret bound for an optimistic least square policy iteration (LSPI) type algorithm. A notable negative result for the online RL setting by Wang et al. (2021) shows that an exponentially large number of samples are needed if we only assume Q^*-realizability and constant sub-optimality gap. Other related works include Ayoub et al. (2020); Jin et al. (2021); Du et al. (2019); Wang et al. (2019), and references therein.

3. Preliminaries

We use $\Delta_{\mathcal{S}}$ to denote the set of probability distributions defined on the set $\mathcal{S}$. Consider an infinite-horizon discounted MDP that is specified by a tuple $(\mathcal{S}, \mathcal{A}, r, P, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the finite action space, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the probability transition kernel, $\rho \in \mathcal{S}$ is the initial state, and $\gamma \in (0, 1)$ is the discount factor. For simplicity, in the main sections of this paper, we assume that the initial state ρ is deterministic and known to the agent. Our algorithm can also be extended to the setting where the initial state is random and the agent is allowed to sample from the initial state distribution. We discuss this extension in Appendix E. Throughout this paper, we write $[N] := \{1, 2, \ldots, N\}$ for any positive integer N and use $\log(\cdot)$ to denote natural logarithm.

A policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ is a mapping from a state to a distribution over actions. We only consider stationary policies, i.e., they do not change according to the time step. The value function $V_\pi(s)$ of a policy is the expected return when we start running the policy π from state s, i.e.,

$$V_\pi(s) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \right],$$

and the state-action value function $Q_\pi(s, a)$, also known as the Q-function, is the expected return following policy π conditioned on $s_0 = s, a_0 = a$, i.e.,

$$Q_\pi(s, a) = \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t), a_{t+1} \sim \pi(\cdot|s_{t+1})} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right].$$

We assume that the agent interacts with a simulator using the local access protocol defined in Definition 1, i.e, for any state s that the agent has visited and any action $a \in \mathcal{A}$, the agent can query the simulator and obtain a sample $s' \sim P(\cdot|s, a)$ and the reward $r(s, a)$.

Our general goal is to find a policy that maximizes the expected return starting from the initial state ρ, i.e., $\max_\pi V_\pi(\rho)$. We let π^* be the optimal policy, $V^*(\cdot) := V_{\pi^*}(\cdot)$, and $Q^*(\cdot, \cdot) := Q_{\pi^*}(\cdot, \cdot)$. We also aim to learn a good policy efficiently, i.e., the query and computational costs should not depend on the size of the state space $\mathcal{S}$, which can be large in many problems.

Linear function approximation Let $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ be a feature map which assigns to each state-action pair a d-dimensional feature vector. For any $(s, a) \in \mathcal{S} \times \mathcal{A}$, the agent can obtain $\phi(s, a)$ with a computational cost of $\mathrm{poly}(d)$. Here, we emphasize that the computation of the feature vectors does not lead to a query cost. Without loss of generality, we impose the following bounded features assumption.

Assumption 2 (Bounded features) *We assume that $\|\phi(s, a)\|_2 \leq 1$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$.*

We consider the following two different assumptions on the linear realizability of the Q-functions:

Assumption 3 (Q_π-realizability) *There exists $b > 0$ such that for every policy π, there exists a weight vector $w_\pi \in \mathbb{R}^d$, $\|w_\pi\|_2 \leq b$, that ensures $Q_\pi(s, a) = w_\pi^\top \phi(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$.*

Assumption 4 (Approximate Q_π-realizability) *There exists $b > 0$ and model misspecification error $\epsilon > 0$ such that for every policy π, there exists a weight vector $w_\pi \in \mathbb{R}^d$, $\|w_\pi\|_2 \leq b$, that ensures $|Q_\pi(s, a) - w_\pi^\top \phi(s, a)| \leq \epsilon$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$.*

4. Algorithm

We first introduce some basic concepts used in our algorithms.

Core set We use a concept called core set. A core set $\mathcal{C}$ is a set of tuples $z = (s, a, \phi(s,a), q) \in \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \times (\mathbb{R} \cup \{\textsf{none}\})$. The first three elements in the tuple denote a state, an action, and the feature vector corresponding to the state-action pair, respectively. The last element $q \in \mathbb{R}$ in the tuple denotes an *estimate* of $Q_\pi(s,a)$ for a policy π. During the algorithm, we may not always have such an estimate, in which case we write $q = \textsf{none}$. For a tuple z, we use z_s, z_a, z_ϕ, and z_q to denote the s, a, ϕ, and q coordinates of z, respectively. We note that in prior works, the core set usually consists of the state-action pairs and their features (Lattimore et al., 2020; Du et al., 2020a; Shariff and Szepesvári, 2020); whereas in this paper, for the convenience of notation, we also have the target values (Q-function estimates) in the core set elements. We denote by $\Phi_\mathcal{C} \in \mathbb{R}^{|\mathcal{C}| \times d}$ the *feature matrix* of all the elements in $\mathcal{C}$, i.e., each row of $\Phi_\mathcal{C}$ is the feature vector of an element in $\mathcal{C}$. Similarly, we define $q_\mathcal{C} \in \mathbb{R}^{|\mathcal{C}|}$ as the vector for the Q_π estimate of all the tuples in $\mathcal{C}$.

Good set It is also useful to introduce a notion of *good set*.

Definition 5 *Given $\lambda, \tau > 0$, and feature matrix $\Phi_\mathcal{C} \in \mathbb{R}^{|\mathcal{C}| \times d}$, the good set $\mathcal{H} \subset \mathbb{R}^d$ is defined as*

$$\mathcal{H} := \{\phi \in \mathbb{R}^d : \phi^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} \phi \le \tau\}.$$

Intuitively, the good set is a set of vectors that are well-covered by the rows of $\Phi_\mathcal{C}$; in other words, these vectors are not closely aligned with the eigenvectors associated with the small eigenvalues of the covariance matrix of all the features in the core set.

As an overview, our algorithm CONFIDENT MC-LSPI works as follows. First, we initialize the core set using the initial state ρ paired with all actions. Then, the algorithm runs least-squares policy iteration (Munos, 2003) to optimize the policy. This means that in each iteration, we estimate the Q-function value for every state-action pair in $\mathcal{C}$ using Monte Carlo rollout with the simulator, and learn a linear function to approximate the Q-function of the rollout policy, and the next policy is chosen to be greedy with respect to this linear function. Our second algorithm CONFIDENT MC-POLITEX works similarly, with the only difference being that instead of using the greedy policy iteration update rule, we use the mirror descent update rule with KL regularization between adjacent rollout policies (Even-Dar et al., 2009; Abbasi-Yadkori et al., 2019). Moreover, in both algorithms, whenever we observe a state-action pair whose feature is not in the good set during Monte Carlo rollout, we add the pair to the core set and restart the policy iteration process. We name the rollout subroutine CONFIDENTROLLOUT. We discuss details in the following.

4.1. Subroutine: CONFIDENTROLLOUT

We first introduce the CONFIDENTROLLOUT subroutine, whose purpose is to estimate $Q_\pi(s_0, a_0)$ for a given state-action pair (s_0, a_0) using Monte Carlo rollouts. During a rollout, for each state s that we encounter and all actions $a \in \mathcal{A}$, the subroutine checks whether the feature vector $\phi(s,a)$ is in the good set. If not, we know that we have discovered a new feature direction, i.e. a direction which is not well aligned with eigenvectors corresponding to the the largest eigenvalues of the covariance matrix of the core features. In this case the subroutine terminates and returns the tuple $(s, a, \phi(s,a), \textsf{none})$ along with the $\textsf{uncertain}$ status. If the algorithm does not discover a new direction, it returns an estimate q of the desired value $Q_\pi(s_0, a_0)$ and the $\textsf{done}$ status. This subroutine

is formally presented in Algorithm 1. We also note that a similar procedure is used in Du et al. (2020b) for a value-iteration-based algorithm.

Algorithm 1 CONFIDENTROLLOUT

Input: number of rollouts m, length of rollout n, rollout policy π, discount γ, initial state s_0, initial action a_0, feature matrix $\Phi_{\mathcal{C}}$, regularization coefficient λ, threshold τ.
for $i = 1, \ldots, m$ **do**
  $s_{i,0} \leftarrow s_0$, $a_{i,0} \leftarrow a_0$, query the simulator, obtain reward $r_{i,0} \leftarrow r(s_{i,0}, a_{i,0})$, and next state $s_{i,1}$.
  for $t = 1, \ldots, n$ **do**
    for $a \in \mathcal{A}$ **do**
      Compute feature $\phi(s_{i,t}, a)$.
      if $\phi(s_{i,t}, a)^\top (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \phi(s_{i,t}, a) > \tau$ **then**
        status $\leftarrow$ uncertain, result $\leftarrow (s_{i,t}, a, \phi(s_{i,t}, a), \text{none})$
        return status, result
      end if
    end for
    Sample $a_{i,t} \sim \pi(\cdot|s_{i,t})$.
    Query the simulator with $s_{i,t}, a_{i,t}$, obtain reward $r_{i,t} \leftarrow r(s_{i,t}, a_{i,t})$, and next state $s_{i,t+1}$.
  end for
end for
status $\leftarrow$ done, result $\leftarrow \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{n} \gamma^t r_{i,t}$
return status, result

4.2. Policy iteration

With the subroutine, now we are ready to present our main algorithms. Both of our algorithms maintain a core set $\mathcal{C}$. We first initialize the core set using the initial state ρ and all actions $a \in \mathcal{A}$. More specifically, we check all the feature vectors $\phi(\rho, a), a \in \mathcal{A}$. If the feature vector is not in the good set of the current core set, we add the tuple $\{(\rho, a, \phi(\rho, a), \text{none})\}$ to the core set. Then we start the policy iteration process. Both algorithms start with an arbitrary initial policy π_0 and run K iterations. Let π_{k-1} be the rollout policy in the k-th iteration. We try to estimate the state-action values for the state-action pairs in $\mathcal{C}$ under the current policy π_{k-1}, i.e., $Q_{\pi_{k-1}}(z_s, z_a)$ for $z \in \mathcal{C}$, using CONFIDENTROLLOUT. In this Q-function estimation procedure, we may encounter two scenarios:

(a) If the rollout subroutine always returns the done status with an estimate of the state-action value, once we finish the estimation for all the state-action pairs in $\mathcal{C}$, we can estimate the Q-function of π_{k-1} using least squares with input features $\Phi_{\mathcal{C}}$ and targets $q_{\mathcal{C}}$ and regularization coefficient λ. Let w_k be the solution to the least squares problem, i.e.,

$$w_k = (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top q_{\mathcal{C}}. \tag{1}$$

Then, for CONFIDENT MC-LSPI, we choose the rollout policy of the next iteration, i.e., π_k, as the greedy policy with respect to the linear function $w_k^\top \phi(s, a)$:

$$\pi_k(a|s) = \mathbb{1}\big(a = \arg\max_{a' \in \mathcal{A}} w_k^\top \phi(s, a')\big). \tag{2}$$

For CONFIDENT MC-POLITEX, we construct a truncated Q-function $Q_{k-1} : \mathcal{S} \times \mathcal{A} \mapsto [0, (1-\gamma)^{-1}]$ using linear function with clipping:

$$Q_{k-1}(s,a) := \Pi_{[0,(1-\gamma)^{-1}]}(w_k^\top \phi(s,a)), \tag{3}$$

where $\Pi_{[a,b]}(x) := \min\{\max\{x,a\},b\}$. The rollout policy of the next iteration is then

$$\pi_k(a|s) \propto \exp\big(\alpha \sum_{j=1}^{k-1} Q_j(s,a)\big), \tag{4}$$

where $\alpha > 0$ is an algorithm parameter.

(b) It could also happen that the CONFIDENTROLLOUT subroutine returns the uncertain status. In this case, we add the state-action pair with new feature direction found by the subroutine to the core set and restart the policy iteration process with the latest core set.

As a final note, for CONFIDENT MC-LSPI, we output the rollout policy of the last iteration π_{K-1}, whereas for CONFIDENT MC-POLITEX, we output a *mixture policy* $\overline{\pi}_K$, which is a policy chosen from $\{\pi_k\}_{k=0}^{K-1}$ uniformly at random. The reason that this algorithm needs to output a mixture policy is that POLITEX (Szepesvári, 2021) uses the regret analysis of expert learning (Cesa-Bianchi and Lugosi, 2006), and to obtain a single output policy, we need to use the standard online-to-batch conversion argument (Cesa-Bianchi et al., 2004). Our algorithms are formally presented in Algorithm 2. In the next section, we present theoretical guarantees for our algorithms.

Algorithm 2 CONFIDENT MC-LSPI / POLITEX

Input: initial state ρ, initial policy π_0, number of iterations K, regularization coefficient λ, threshold τ, discount γ, number of rollouts m, length of rollout n, POLITEX parameter α.
$\mathcal{C} \leftarrow \emptyset$ *// Initialize core set.*
for $a \in \mathcal{A}$ **do**
  if $\mathcal{C} = \emptyset$ or $\phi(\rho,a)^\top(\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1}\phi(\rho,a) > \tau$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\rho, a, \phi(\rho,a), \mathsf{none})\}$
  end if
end for
$z_q \leftarrow \mathsf{none},\ \forall z \in \mathcal{C}$ *// Policy iteration starts.* $(*)$
for $k = 1, \ldots, K$ **do**
  for $z \in \mathcal{C}$ **do**
    $\mathsf{status}, \mathsf{result} \leftarrow$ CONFIDENTROLLOUT$(m, n, \pi_{k-1}, \gamma, z_s, z_a, \Phi_\mathcal{C}, \lambda, \tau)$
    if $\mathsf{status} = \mathsf{done}$, **then** $z_q \leftarrow \mathsf{result}$; **else** $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{result}\}$ and **goto** line $(*)$
  end for
  $w_k \leftarrow (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} \Phi_\mathcal{C}^\top q_\mathcal{C}$; $Q_{k-1}(s,a) \leftarrow \Pi_{[0,(1-\gamma)^{-1}]}(w_k^\top \phi(s,a))$ (POLITEX only)
  $\pi_k(a|s) \leftarrow \begin{cases} \mathbb{1}\big(a = \arg\max_{a' \in \mathcal{A}} w_k^\top \phi(s,a')\big), & \text{LSPI} \\ \exp\big(\alpha \sum_{j=1}^{k-1} Q_j(s,a)\big) / \sum_{a' \in \mathcal{A}} \exp\big(\alpha \sum_{j=1}^{k-1} Q_j(s,a')\big) & \text{POLITEX} \end{cases}$
end for
return w_{K-1} for LSPI, or $\overline{\pi}_K \sim \mathrm{Unif}\{\pi_k\}_{k=0}^{K-1}$ for POLITEX.

5. Theoretical guarantees

In this section, we present theoretical guarantees for our algorithms. First, we have the following main result for CONFIDENT MC-LSPI.

Theorem 6 (Main result for CONFIDENT MC-LSPI) *If Assumption 3 holds, then for an arbitrarily small $\kappa > 0$, by choosing $\tau = 1$, $\lambda = \frac{\kappa^2(1-\gamma)^4}{1024b^2}$, $n = \frac{3}{1-\gamma}\log(\frac{4(1+\log(1+\lambda^{-1})d)}{\kappa(1-\gamma)})$, $K = 2 + \frac{2}{1-\gamma}\log(\frac{3}{\kappa(1-\gamma)})$, $m = 4096\frac{d(1+\log(1+\lambda^{-1}))}{\kappa^2(1-\gamma)^6}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we have with probability at least $1-\delta$, the policy π_{K-1} that* CONFIDENT MC-LSPI *outputs satisfies*

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \kappa.$$

Moreover, the query and computational costs for the algorithm are $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}), \log(b))$ *and* $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, |\mathcal{A}|, \log(\frac{1}{\delta}), \log(b))$*, respectively.*

Alternatively, if Assumption 4 holds, then by choosing $\tau = 1$, $\lambda = \frac{\epsilon^2 d}{b^2}$, $n = \frac{1}{1-\gamma}\log(\frac{1}{\epsilon(1-\gamma)})$, $K = 2 + \frac{1}{1-\gamma}\log\left(\frac{1}{\epsilon\sqrt{d}}\right)$, $m = \frac{1}{\epsilon^2(1-\gamma)^2}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we have with probability at least $1-\delta$, the policy π_{K-1} that CONFIDENT MC-LSPI *outputs satisfies*

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \frac{74\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1})).$$

Moreover, the query and computational costs for the algorithm are $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(b))$ *and* $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, |\mathcal{A}|, \log(\frac{1}{\delta}), \log(b))$*, respectively.*

We prove Theorem 6 in Appendix B. For CONFIDENT MC-POLITEX, since we output a mixture policy, we prove guarantees for the *expected value of the mixture policy*, i.e., $V_{\overline{\pi}_K} := \frac{1}{K}\sum_{k=0}^{K-1} V_{\pi_k}$. We have the following result.

Theorem 7 (Main result for CONFIDENT MC-POLITEX) *If Assumption 3 holds, then for an arbitrarily small $\kappa > 0$, by choosing $\tau = 1$, $\alpha = (1-\gamma)\sqrt{\frac{2\log(|\mathcal{A}|)}{K}}$, $\lambda = \frac{\kappa^2(1-\gamma)^2}{256b^2}$, $K = \frac{32\log(|\mathcal{A}|)}{\kappa^2(1-\gamma)^4}$, $n = \frac{1}{1-\gamma}\log(\frac{32\sqrt{d}(1+\log(1+\lambda^{-1}))}{(1-\gamma)^2\kappa})$, and $m = 1024\frac{d(1+\log(1+\lambda^{-1}))}{\kappa^2(1-\gamma)^4}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we have with probability at least $1-\delta$, the mixture policy $\overline{\pi}_K$ that* CONFIDENT MC-POLITEX *outputs satisfies*

$$V^*(\rho) - V_{\overline{\pi}_K}(\rho) \leq \kappa.$$

Moreover, the query and computational costs for the algorithm are $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}), \log(b))$ *and* $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, |\mathcal{A}|, \log(\frac{1}{\delta}), \log(b))$*, respectively.*

Alternatively, if Assumption 4 holds, then by choosing $\tau = 1$, $\alpha = (1-\gamma)\sqrt{\frac{2\log(|\mathcal{A}|)}{K}}$, $\lambda = \frac{\epsilon^2 d}{b^2}$, $K = \frac{2\log(|\mathcal{A}|)}{\epsilon^2 d(1-\gamma)^2}$, $n = \frac{1}{1-\gamma}\log(\frac{1}{\epsilon(1-\gamma)})$, and $m = \frac{1}{\epsilon^2(1-\gamma)^2}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we have with probability at least $1-\delta$, the mixture policy $\overline{\pi}_K$ that CONFIDENT MC-POLITEX *outputs satisfies*

$$V^*(\rho) - V_{\overline{\pi}_K}(\rho) \leq \frac{42\epsilon\sqrt{d}}{1-\gamma}(1 + \log(1 + b^2\epsilon^{-2}b^{-1})).$$

Moreover, the query and computational costs for the algorithm are $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(b))$ *and* $\mathrm{poly}(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, |\mathcal{A}|, \log(\frac{1}{\delta}), \log(b))$*, respectively.*

We prove Theorem 7 in Appendix D. Here, we first discuss the query and computational costs of both algorithms and then provide a sketch of our proof.

Query and computational costs In our analysis, we say that we start a new *loop* whenever we start (or restart) the policy iteration process, i.e., going to line $(*)$ in Algorithm 2. By definition, when we start a new loop, the size of the core set $\mathcal{C}$ is increased by 1. First, in Lemma 8 below, we show that the size of the core set will never exceed $C_{\max} = \tilde{\mathcal{O}}(d)$. Therefore, the total number of loops is at most $C_{\max}$. In each loop, we run K policy iterations; in each iteration, we run Algorithm 1 from at most $C_{\max}$ points from the core set; and each time when we run Algorithm 1, we query the simulator at most $\mathcal{O}(mn)$ times. Thus, for both algorithms, the total number of queries that we make is at most $C_{\max}^2 Kmn$. Therefore, using the parameter choice in Theorems 6 and 7 and omitting logarithmic factors, we can obtain the query costs of CONFIDENT MC-LSPI and POLITEX in Table 2. As we can see, when $\epsilon = 0$ or $\epsilon \neq 0$ but $\epsilon = o(1/\sqrt{d})$ (the regime we care about in this paper), the query cost of CONFIDENT MC-LSPI is lower than POLITEX. As for computational cost, since our policy improvement steps only involve matrix multiplication and matrix inversion, the computational cost is also polynomial in the aforementioned factors. One thing to notice is that during the rollout process, in each step, the agent needs to compute the features of a state paired with all actions, and thus the computational cost linearly depends on $|\mathcal{A}|$; on the contrary the query cost does not depend on $|\mathcal{A}|$ since in each step the agent only needs to query the simulator with the action sampled according to the policy.

Sub-optimality We also note that when Assumption 4 holds, i.e., $\epsilon \neq 0$, the sub-optimality of the output policy is $\tilde{\mathcal{O}}(\frac{\epsilon\sqrt{d}}{(1-\gamma)^2})$ for LSPI and $\tilde{\mathcal{O}}(\frac{\epsilon\sqrt{d}}{1-\gamma})$ for POLITEX. Therefore, in the presence of a model misspecification error, CONFIDENT MC-POLITEX can achieve a better final sub-optimality than CONFIDENT MC-LSPI, although it's query cost is higher.

Table 2: Comparison of CONFIDENT MC-LSPI and POLITEX

	Query ($\epsilon = 0$)	Query ($\epsilon \neq 0$)	Sub-optimality ($\epsilon \neq 0$)
LSPI	$\tilde{\mathcal{O}}(\frac{d^3}{\kappa^2(1-\gamma)^8})$	$\tilde{\mathcal{O}}(\frac{d^2}{\epsilon^2(1-\gamma)^4})$	$\tilde{\mathcal{O}}(\frac{\epsilon\sqrt{d}}{(1-\gamma)^2})$
POLITEX	$\tilde{\mathcal{O}}(\frac{d^3}{\kappa^4(1-\gamma)^9})$	$\tilde{\mathcal{O}}(\frac{d}{\epsilon^4(1-\gamma)^5})$	$\tilde{\mathcal{O}}(\frac{\epsilon\sqrt{d}}{1-\gamma})$

Proof sketch We now discuss our proof strategy, focusing on LSPI for simplicity.

Step 1: Bound the size of the core set The first step is to show that our algorithm will terminate. This is equivalent to showing that the size of the core set $\mathcal{C}$ will not exceed certain finite quantity, since whenever we receive the uncertain status from CONFIDENTROLLOUT, we increase the size of the core set by 1, go back to line $(*)$ in Algorithm 2, and start a new loop. The following lemma shows that the size of the core set is always bounded, and thus the algorithm will always terminate.

Lemma 8 *Under Assumption 2, the size of the core set $\mathcal{C}$ will not exceed*

$$C_{\max} := \frac{e}{e-1}\frac{1+\tau}{\tau} d \left(\log(1+\frac{1}{\tau}) + \log(1+\frac{1}{\lambda}) \right).$$

This result first appears in Russo and Van Roy (2013) as the eluder dimension of linear function class. We present the proof of this lemma in Appendix A for completeness.

Step 2: Virtual policy iteration The next step is to analyze the gap between the value of the optimal policy and the policy π parameterized by the vector w_{K-1} that the algorithm outputs in the final loop, i.e., $V^*(\rho) - V_{\pi_{K-1}}(\rho)$. For ease of exposition, here we only consider the case of deterministic probability transition kernel P. Our full proof in Appendix B considers general stochastic dynamics.

To analyze our algorithm, we note that for approximate policy iteration (API) algorithms, if in every iteration (say the k-th iteration), we have an approximate Q-function that is close to the true Q-function of the rollout policy (say π_{k-1}) in ℓ_∞ norm, i.e., $\|Q_{k-1} - Q_{\pi_{k-1}}\|_\infty \leq \eta$, then existing results (Munos, 2003; Farahmand et al., 2010) ensure that we can learn a good policy if in every iteration we choose the new policy to be greedy with respect to the approximate Q-function. However, since we only have local access to the simulator, we cannot have such ℓ_∞ guarantee. In fact, as we show in the proof, we can only ensure that when $\phi(s,a)$ is in the good set $\mathcal{H}$, our linear function approximation is accurate, i.e., $|Q_{k-1}(s,a) - Q_{\pi_{k-1}}(s,a)| \leq \eta$ where $Q_{k-1}(s,a) = w_k^\top \phi(s,a)$. To overcome the lack of ℓ_∞ guarantee, we introduce the notion of *virtual policy iteration* algorithm. In the virtual algorithm, we start with the same initial policy $\tilde{\pi}_0 = \pi_0$. In the k-th iteration of the virtual algorithm, we assume that we have access to the true Q-function of the rollout policy $\tilde{\pi}_{k-1}$ when $\phi(s,a) \notin \mathcal{H}$, and construct

$$\tilde{Q}_{k-1}(s,a) = \begin{cases} \tilde{w}_k^\top \phi(s,a) & \text{if } \phi(s,a) \in \mathcal{H} \\ Q_{\tilde{\pi}_{k-1}}(s,a) & \text{otherwise,} \end{cases}$$

where $\tilde{w}_k$ is the linear coefficient that we learn in the virtual algorithm in the same way as in Eq. (1). Then $\tilde{\pi}_k$ is chosen to be greedy with respect to $\tilde{Q}_{k-1}(s,a)$. In this way, we can ensure that $\tilde{Q}_{k-1}(s,a)$ is close to the true Q-function $Q_{\tilde{\pi}_{k-1}}(s,a)$ in ℓ_∞ norm and thus the output policy, say $\tilde{\pi}_{K-1}$, of the virtual algorithm is good in the sense that $V^*(\rho) - V_{\tilde{\pi}_{K-1}}(\rho)$ is small.

To connect the output policy of the virtual algorithm and our actual algorithm, we note that by definition, in the final loop of our algorithm, in any iteration, for any state s that the agent visits in CONFIDENTROLLOUT, and any action $a \in \mathcal{A}$, we have that $\phi(s,a) \in \mathcal{H}$ since the subroutine never returns uncertain status. Further, because the initial state, probability transition kernel, and the policies are all deterministic, we know that the rollout trajectories of the virtual algorithm and our actual algorithm are always the same in the final loop (the virtual algorithm does not get a chance to use the true Q-function $Q_{\tilde{\pi}_{k-1}}$). With rollout length n, we know that when we start with state ρ, the output of the virtual algorithm $\tilde{\pi}_{K-1}$ and our actual algorithm π_{K-1} take exactly the same actions for n steps, and thus $|V_{\pi_{K-1}}(\rho) - V_{\tilde{\pi}_{K-1}}(\rho)| \leq \frac{\gamma^{n+1}}{1-\gamma}$, which implies that $V^*(\rho) - V_{\pi_{K-1}}(\rho)$ is small. To extend this argument to the setting with stochastic transitions, we need to use a coupling argument which we elaborate in the Appendix.

6. Conclusion

We propose the CONFIDENT MC-LSPI and CONFIDENT MC-POLITEX algorithms, for local planning with linear function approximation. Under the assumption that the Q-functions of all policies are linear in some features of the state-action pairs, we show that our algorithm is query and computationally efficient. We introduce a novel analysis technique based on a virtual policy iteration algorithm, which can be used to leverage existing guarantees on approximate policy iteration with ℓ_∞-bounded evaluation error. We use this technique to show that our algorithm can learn the opti-

mal policy for the given initial state even only with local access to the simulator. Future directions include extending our analysis technique to broader settings.

Acknowledgments

The authors would like to thank Gellért Weisz for helpful comments.

References

Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellért Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pages 3692–3702. PMLR, 2019.

Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. PC-PG: Policy cover directed exploration for provable policy gradient learning. *arXiv preprint arXiv:2007.08459*, 2020a.

Alekh Agarwal, Sham Kakade, and Lin F Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory*, pages 67–83. PMLR, 2020b.

Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3): 325–349, 2013.

Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *International Conference on Machine Learning*, pages 1283–1294. PMLR, 2020.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Simon S Du, Yuping Luo, Ruosong Wang, and Hanrui Zhang. Provably efficient Q-learning with function approximation via distribution shift error checking oracle. *arXiv preprint arXiv:1906.06321*, 2019.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020a.

Simon S Du, Jason D Lee, Gaurav Mahajan, and Ruosong Wang. Agnostic Q-learning with function approximation in deterministic systems: Tight bounds on approximation error and sample complexity. *arXiv preprint arXiv:2002.07125*, 2020b.

Simon S Du, Sham M Kakade, Jason D Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear classes: A structural framework for provable generalization in RL. *arXiv preprint arXiv:2103.10897*, 2021.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. Online Markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

Amir Massoud Farahmand, Rémi Munos, and Csaba Szepesvári. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems*, 2010.

Botao Hao, Nevena Lazic, Yasin Abbasi-Yadkori, Pooria Joulani, and Csaba Szepesvári. Adaptive approximate policy iteration. In *International Conference on Artificial Intelligence and Statistics*, pages 523–531. PMLR, 2021.

David A Harville. Matrix algebra from a statistician's perspective, 1998.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2017.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of RL problems, and sample-efficient algorithms. *arXiv preprint arXiv:2102.00815*, 2021.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Michael Kearns and Satinder Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. *Advances in Neural Information Processing Systems*, pages 996–1002, 1999.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine learning*, 49(2):193–208, 2002.

Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in RL with a generative model. In *International Conference on Machine Learning*, pages 5662–5670. PMLR, 2020.

Nevena Lazic, Dong Yin, Yasin Abbasi-Yadkori, and Csaba Szepesvari. Improved regret bound and experience replay in regularized policy iteration. *arXiv preprint arXiv:2102.12611*, 2021.

Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Breaking the sample size barrier in model-based reinforcement learning with a generative model. *Advances in Neural Information Processing Systems*, 33, 2020.

Gen Li, Yuxin Chen, Yuejie Chi, Yuantao Gu, and Yuting Wei. Sample-efficient reinforcement learning is feasible for linearly realizable MDPs with limited revisiting. *arXiv preprint arXiv:2105.08024*, 2021.

Rémi Munos. Error bounds for approximate policy iteration. In *International Conference on Machine Learning*, volume 3, pages 560–567, 2003.

Rémi Munos. From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 2014.

Daniel Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *Advances in Neural Information Processing Systems*, pages 2256–2264. Citeseer, 2013.

Roshan Shariff and Csaba Szepesvári. Efficient planning in large MDPs with weak linear function approximation. *arXiv preprint arXiv:2007.06184*, 2020.

Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving Markov decision processes with a generative model. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5192–5202, 2018.

Satinder P Singh and Richard C Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

Csaba Szepesvári. RL Theory lecture notes: POLITEX. https://rltheory.github.io/lecture-notes/planning-in-mdps/lec14/, 2021.

Michael J Todd. *Minimum-volume ellipsoids: Theory and algorithms*. SIAM, 2016.

Benjamin Van Roy and Shi Dong. Comments on the Du-Kakade-Wang-Yang lower bounds. *arXiv preprint arXiv:1911.07910*, 2019.

Yining Wang, Ruosong Wang, Simon S Du, and Akshay Krishnamurthy. Optimism in reinforcement learning with generalized linear function approximation. *arXiv preprint arXiv:1912.04136*, 2019.

Yuanhao Wang, Ruosong Wang, and Sham M Kakade. An exponential lower bound for linearly-realizable MDPs with constant suboptimality gap. *arXiv preprint arXiv:2103.12690*, 2021.

Chen-Yu Wei, Mehdi Jafarnia Jahromi, Haipeng Luo, and Rahul Jain. Learning infinite-horizon average-reward MDPs with linear function approximation. In *International Conference on Artificial Intelligence and Statistics*, pages 3007–3015. PMLR, 2021.

Gellert Weisz, Philip Amortila, Barnabás Janzer, Yasin Abbasi-Yadkori, Nan Jiang, and Csaba Szepesvári. On query-efficient planning in MDPs under linear realizability of the optimal state-value function. *arXiv preprint arXiv:2102.02049*, 2021a.

Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in MDPs with linearly-realizable optimal action-value functions. In *Algorithmic Learning Theory*, pages 1237–1264. PMLR, 2021b.

Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. *Advances in Neural Information Processing Systems*, 26, 2013.

Lin Yang and Mengdi Wang. Sample-optimal parametric Q-learning using linearly additive features. In *International Conference on Machine Learning*, pages 6995–7004. PMLR, 2019.

Lin Yang and Mengdi Wang. Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound. *International Conference on Machine Learning*, 2020.

Andrea Zanette, Alessandro Lazaric, Mykel J Kochenderfer, and Emma Brunskill. Limiting extrapolation in linear approximate value iteration. *Advances in Neural Information Processing Systems*, 32:5615–5624, 2019.

Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent Bellman error. In *International Conference on Machine Learning*, pages 10978–10989. PMLR, 2020.

Andrea Zanette, Ching-An Cheng, and Alekh Agarwal. Cautiously optimistic policy optimization and exploration with linear function approximation. In *Conference on Learning Theory (COLT)*, 2021.

Dongruo Zhou, Jiafan He, and Quanquan Gu. Provably efficient reinforcement learning for discounted MDPs with feature mapping. *arXiv preprint arXiv:2006.13165*, 2020.

Appendix

Appendix A. Proof of Lemma 8

This proof essentially follows the proof of the upper bound for the eluder dimension of a linear function class in Russo and Van Roy (2013). We present the proof here for completeness.

We restate the core set construction process in the following way with slightly different notation. We begin with $\Phi_0 = 0$. In the t-th step, we have a core set with feature matrix $\Phi_{t-1} \in \mathbb{R}^{(t-1)\times d}$. Suppose that we can find $\phi_t \in \mathbb{R}^d$, $\|\phi_t\|_2 \le 1$, such that

$$\phi_t^\top (\Phi_{t-1}^\top \Phi_{t-1} + \lambda I)^{-1} \phi_t > \tau, \tag{5}$$

then we let $\Phi_t := [\Phi_{t-1}^\top \ \ \phi_t]^\top \in \mathbb{R}^{t\times d}$, i.e., we add a row at the bottom of Φ_{t-1}. If we cannot find such ϕ_t, we terminate this process. We define $\Sigma_t := \Phi_t^\top \Phi_t + \lambda I$. It is easy to see that $\Sigma_0 = \lambda I$ and $\Sigma_t = \Sigma_{t-1} + \phi_t \phi_t^\top$.

According to matrix determinant lemma (Harville, 1998), we have

$$\begin{aligned} \det(\Sigma_t) &= (1 + \phi_t^\top \Sigma_{t-1}^{-1} \phi_t) \det(\Sigma_{t-1}) > (1+\tau)\det(\Sigma_{t-1}) \\ &> \cdots > (1+\tau)^t \det(\Sigma_0) = (1+\tau)^t \lambda^d, \end{aligned} \tag{6}$$

where the inequality is due to (5). Since $\det(\Sigma_t)$ is the product of all the eigenvalues of Σ_t, according to the AM-GM inequality, we have

$$\det(\Sigma_t) \leq \left(\frac{\mathrm{tr}(\Sigma_t)}{d}\right)^d = \left(\frac{\mathrm{tr}(\sum_{i=1}^t \phi_i \phi_i^\top) + \mathrm{tr}(\lambda I)}{d}\right)^d \leq (\frac{t}{d} + \lambda)^d, \tag{7}$$

where in the second inequality we use the fact that $\|\phi_i\|_2 \leq 1$. Combining (6) and (7), we know that t must satisfy

$$(1+\tau)^t \lambda^d < (\frac{t}{d} + \lambda)^d,$$

which is equivalent to

$$(1+\tau)^{\frac{t}{d}} < \frac{t}{\lambda d} + 1. \tag{8}$$

We note that if $t \leq d$, the result of the size of the core set in Lemma 8 automatically holds. Thus, we only consider the situation here $t > d$. In this case, the condition (8) implies

$$\begin{aligned}\frac{t}{d}\log(1+\tau) < \log(1+\frac{t}{\lambda d}) < \log(\frac{t}{d}(1+\frac{1}{\lambda})) &= \log(\frac{t}{d}) + \log(1+\frac{1}{\lambda}) \\ &= \log\left(\frac{t\tau}{d(1+\tau)}\right) + \log(\frac{1+\tau}{\tau}) + \log(1+\frac{1}{\lambda}).\end{aligned} \tag{9}$$

Using the fact that for any $x > 0$, $\log(1+x) > \frac{x}{1+x}$, and that for any $x > 0$, $\log(x) \leq \frac{x}{e}$, we obtain

$$\frac{t\tau}{d(1+\tau)} < \frac{t\tau}{ed(1+\tau)} + \log(\frac{1+\tau}{\tau}) + \log(1+\frac{1}{\lambda}), \tag{10}$$

which implies

$$t < \frac{e}{e-1}\frac{1+\tau}{\tau} d \left(\log(1+\frac{1}{\tau}) + \log(1+\frac{1}{\lambda})\right).$$

Appendix B. Proof of Theorem 6

In this proof, we say that we start a new *loop* whenever we start (or restart) the policy iteration process, i.e., going to line $(*)$ in Algorithm 2. In each loop, we have at most K iterations of policy iteration steps. By definition, we also know that when we start a new loop, the size of the core set $\mathcal{C}$ increases by 1 compared with the previous loop. We first introduce the notion of virtual policy iteration algorithm. This virtual algorithm is designed to leverage the existing results on approximate policy iteration with ℓ_∞ bounded error in the approximate Q-functions (Munos, 2003; Farahmand et al., 2010). We first present the details of the virtual algorithm, and then provide performance guarantees for the main algorithm.

B.1. Virtual approximate policy iteration with coupling

The virtual policy iteration algorithm is a virtual algorithm that we use for the purpose of proof. It is a version of approximate policy iteration (API) with a simulator. An important factor is that the

simulators of the virtual algorithm and the main algorithm need to be *coupled*, which we explain in this section.

The virtual algorithm is defined as follows. Unlike the main algorithm, the virtual algorithm runs exactly $C_{\max}$ loops, where $C_{\max}$ is the upper bound for the size of the core set defined in Lemma 8. In the virtual algorithm, we let the initial policy be the same as the main algorithm, i.e., $\tilde{\pi}_0 = \pi_0$. Unlike the main algorithm, the virtual algorithm runs exactly K iterations of policy iteration. In the k-th iteration ($k \geq 1$), the virtual algorithm runs rollouts from each element in the core set $\mathcal{C}$ (we will discuss how the virtual algorithm constructs the core set later) with $\tilde{\pi}_{k-1}$ with a simulator where $\tilde{\pi}_{k-1}$ is in the form of Eq. (13) ($\tilde{Q}_{k-1}$ will be defined once we present the details of the virtual algorithm).

We now describe the rollout process of the virtual algorithm. We still use a subroutine similar to CONFIDENTROLLOUT. The simulator of the virtual algorithm can still generate samples of next state given a state-action pair according to the probability transition kernel P. The major difference from the main algorithm is that during the rollout process, when we find a state-action pair whose feature is outside of the good set $\mathcal{H}$ (defined in Definition 5), i.e., (s, a) such that $\phi(s, a)^\top (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)\phi(s, a) > \tau$, we do not terminate the subroutine, instead we record this state-action pair along with its feature (we call it the *recorded element*), and then keep running the rollout process using $\tilde{\pi}_{k-1}$. Two situations can occur at the end of each loop: 1) We did not record any element, in which case we use the same core set $\mathcal{C}$ in the next loop, and 2) We have at least one recorded element in a particular loop, in which case we add the *first* element to the core set and discard any other recorded elements. In other words, in each loop of the virtual algorithm, we find the first state-action pair (if any) whose feature is outside of the good set and add this pair to the core set. Another difference from the main algorithm is that in the virtual algorithm, we do not end the rollout subroutine when we identify an uncertain state-action pair, and as a result, the rollout subroutine in the virtual algorithm *always returns* an estimation of the Q-function.

We now proceed to present the virtual policy iteration process. In the k-th iteration, the virtual algorithm runs m trajectories of n-step rollout using the policy $\tilde{\pi}_{k-1}$ from each element $z \in \mathcal{C}$, obtains the empirical average of the discounted return z_q in the same way as in Algorithm 1. Then we concatenate them, obtain the vector $\tilde{q}_{\mathcal{C}}$, and compute

$$\tilde{w}_k = (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top \tilde{q}_{\mathcal{C}}. \tag{11}$$

We use the notion of good set $\mathcal{H}$ defined in Definition 5, and define the *virtual Q-function* as follows:

$$\tilde{Q}_{k-1}(s, a) := \begin{cases} \tilde{w}_k^\top \phi(s, a), & \phi(s, a) \in \mathcal{H}, \\ Q_{\tilde{\pi}_{k-1}}(s, a), & \phi(s, a) \notin \mathcal{H}, \end{cases} \tag{12}$$

by assuming the access to the true Q-function $Q_{\tilde{\pi}_{k-1}}(s, a)$. The next policy $\tilde{\pi}_k$ is defined as the greedy policy with respect to $\tilde{Q}_{k-1}(s, a)$, i.e.,

$$\tilde{\pi}_k(a|s) = \mathbb{1}\left(a = \arg\max_{a' \in \mathcal{A}} \tilde{Q}_{k-1}(s, a'))\right). \tag{13}$$

Recall that for the main algorithm, once we learn the parameter vector w_k, the next policy π_k is greedy with respect to the linear function $w_k^\top \phi(s, a)$, i.e.,

$$\pi_k(a|s) = \mathbb{1}\left(a = \arg\max_{a' \in \mathcal{A}} w_k^\top \phi(s, a'))\right).$$

For comparison, the key difference is that when we observe a feature vector $\phi(s,a)$ that is not in the good set $\mathcal{H}$, our actual algorithm terminates the rollout and returns the state-action pair with the new direction, whereas the virtual algorithm uses the true Q-function of the state-action pair.

Coupling The major remaining issue now is how the main algorithm is connected to the virtual algorithm. We describe this connection with a coupling argument. In a particular loop, for any positive integer N, when the virtual algorithm makes its N-th query in the k-th iteration to the virtual simulator with a state-action pair, say $(s_{\text{virtual}}, a_{\text{virtual}})$, if the main algorithm has not returned due to encountering an uncertain state-action pair, we assume that at the same time the main algorithm also makes its N-th query to the simulator, with a state-action pair, say $(s_{\text{main}}, a_{\text{main}})$. We let the two simulators be *coupled*: When they are queried with the same pair, i.e., $(s_{\text{main}}, a_{\text{main}}) = (s_{\text{virtual}}, a_{\text{virtual}})$, the next states that they return are also the same. In other words, the simulator for the main algorithm samples $s'_{\text{main}} \sim P(\cdot|s_{\text{main}}, a_{\text{main}})$, and the virtual algorithm samples $s'_{\text{virtual}} \sim P(\cdot|s_{\text{virtual}}, a_{\text{virtual}})$, and s'_{main} and s'_{virtual} satisfy the joint distribution such that $\mathbb{P}\left(s'_{\text{main}} = s'_{\text{virtual}}\right) = 1$. In the cases where $(s_{\text{main}}, a_{\text{main}}) \neq (s_{\text{virtual}}, a_{\text{virtual}})$ or the main algorithm has already returned due to the discovery of a new feature direction, the virtual algorithm samples from P independently from the main algorithm. Note that this setup guarantees that both the virtual algorithm and the main algorithm have valid simulators which can sample from the same probability transition kernel P.

There are a few direct consequences of this coupling design. First, since the virtual and main algorithms start with the same initial core set elements (constructed using the initial state), we know that in any loop, when starting from the same core set element z, both algorithms will have *exactly the same rollout trajectories* until the main algorithm identifies an uncertain state-action pair and returns. This is due to the coupling of the simulators and the fact that within the good set $\mathcal{H}$, the policies for the main algorithm and the virtual algorithm take the same action. Later, we will discuss this point more in Lemma 13. Second, the core set elements that the virtual and main algorithms use are exactly the same for any loop. This is because when the main algorithm identifies an uncertain state-action pair, it adds it to the core set and start a new loop, and the virtual algorithm also only adds the *first recorded element* to the core set. Since the simulators are the coupled, the first uncertain state-action pair that they encounter will be the same, meaning that both algorithms always add the same element to the core set, until the main algorithm finishes its final loop. We note that the core set elements on our algorithm are stored as ordered list so the virtual and main algorithm always run rollouts with the same ordering of the core set elements. Another observation is that while the virtual algorithm has a deterministic number of loops $C_{\max}$, the total number of loops that the main algorithms may run is a random variable whose value cannot exceed $C_{\max}$.

The next steps of the proof are the following:

- We show that in each loop, with high probability, the virtual algorithm proceeds as an approximate policy iteration algorithm with a bounded ℓ_∞ error in the approximate Q-function. Thus the virtual algorithm produces a good policy at the end of each loop. Then, since by Lemma 8, we have at most

$$C_{\max} := \frac{e}{e-1}\frac{1+\tau}{\tau} d \left(\log(1+\frac{1}{\tau}) + \log(1+\frac{1}{\lambda}) \right) \tag{14}$$

  loops, with a union bound, we know that with high probability, the virtual algorithm produces a good policy in all the loops.

- We show that due to the coupling argument, the output parameter vector in the main and the virtual algorithms, i.e., w_{K-1} and $\tilde{w}_{K-1}$ in the final loop are the same. This leads to the conclusion that with the same initial state ρ, the value of the outputs of the main algorithm and the virtual algorithm are close, and thus the main algorithm also outputs a good policy.

We prove these two points in Sections B.2 and B.3, respectively.

B.2. Analysis of the virtual algorithm

Throughout this section, we will consider a fixed loop of the virtual algorithm, say the ℓ-th loop. We assume that at the beginning of this loop, the virtual algorithm has a core set $\mathcal{C}_\ell$. Notice that $\mathcal{C}_\ell$ is a random variable that only depends on the filtration of the first $\ell - 1$ loops. In this section, we will first condition on the filtration of all the first $\ell - 1$ loops and only consider the filtration of the ℓ-th loop. Thus we will first treat $\mathcal{C}_\ell$ as a deterministic quantity. For simplicity, we write $\mathcal{C} := \mathcal{C}_\ell$.

Consider the k-th iteration of a particular loop of the virtual algorithm with core set $\mathcal{C}$. We would like to bound $\|\tilde{Q}_{k-1} - Q_{\tilde{\pi}_{k-1}}\|_\infty$. First, we have the following lemma for the accuracy of the Q-function for any element in the core set. To simplify notation, in this lemma, we omit the subscript and use π to denote a policy that we run rollout with in an arbitrary iteration of the virtual algorithm.

Lemma 9 *Let π be a policy that we run rollout with in an iteration of the virtual algorithm. Then, for any element $z \in \mathcal{C}$ and any $\theta > 0$, we have with probability at least $1 - 2\exp(-2\theta^2(1-\gamma)^2 m)$,*

$$|z_q - Q_\pi(z_s, z_a)| \leq \frac{\gamma^{n+1}}{1-\gamma} + \theta. \tag{15}$$

Proof By the definition of $Q_\pi(z_s, z_a)$:

$$Q_\pi(z_s, z_a) = \mathbb{E}_{s_{t+1}\sim P(\cdot|s_t,a_t), a_{t+1}\sim\pi(\cdot|s_{t+1})}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = z_s, a_0 = z_a\right],$$

and define the n-step truncated Q-function:

$$Q_\pi^n(z_s, z_a) = \mathbb{E}_{s_{t+1}\sim P(\cdot|s_t,a_t), a_{t+1}\sim\pi(\cdot|s_{t+1})}\left[\sum_{t=0}^{n} \gamma^t r(s_t, a_t) \mid s_0 = z_s, a_0 = z_a\right].$$

Then we have $|Q_\pi^n(s,a) - Q_\pi(s,a)| \leq \frac{\gamma^{n+1}}{1-\gamma}$. Moreover, the Q-function estimate z_q is an average of m independent and unbiased estimates of $Q_\pi^n(s,a)$, which are all bounded in $[0, 1/(1-\gamma)]$. By Hoeffding's inequality we have with probability at least $1 - 2\exp(-2\theta^2(1-\gamma)^2 m)$, $|z_q - Q_\pi^n(s,a)| \leq \theta$, which completes the proof. ■

By a union bound over the $|\mathcal{C}|$ elements in the core set, we know that

$$\mathbb{P}\left(\forall\, z \in \mathcal{C}, |z_q - Q_{\tilde{\pi}_{k-1}}(z_s, z_a)| \leq \frac{\gamma^{n+1}}{1-\gamma} + \theta\right) \geq 1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 m). \tag{16}$$

The following lemma provides a bound on $|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)|, \forall\, (s,a)$ such that $\phi(s,a) \in \mathcal{H}$.

Lemma 10 *Suppose that Assumption 4 holds. Then, with probability at least*

$$1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 m),$$

for any (s,a) pair such that $\phi(s,a) \in \mathcal{H}$, we have

$$|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)| \leq b\sqrt{\lambda\tau} + \Big(\epsilon + \frac{\gamma^{n+1}}{1-\gamma} + \theta\Big)\sqrt{\tau C_{\max}} + \epsilon := \eta. \tag{17}$$

We prove this lemma in Appendix C. Since when $\phi(s,a) \notin \mathcal{H}$, $\tilde{Q}_{k-1}(s,a) = Q_{\tilde{\pi}_{k-1}}(s,a)$, we know that $\|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)\|_\infty \leq \eta$. With another union bound over the K iterations, we know that with probability at least

$$1 - 2KC_{\max}\exp(-2\theta^2(1-\gamma)^2 m),$$

the virtual algorithm is an approximate policy iteration algorithm with ℓ_∞ bound η for the approximation error on the Q-functions. We use the following results for API, which is a direct consequence of the results in Munos (2003); Farahmand et al. (2010), and is also stated in Lattimore et al. (2020).

Lemma 11 *Suppose that we run K approximate policy iterations and generate a sequence of policies $\pi_0, \pi_1, \ldots, \pi_K$. Suppose that for every $k = 1, 2, \ldots, K$, in the k-th iteration, we obtain a function $\tilde{Q}_{k-1}$ such that, $\|\tilde{Q}_{k-1} - Q_{\pi_{k-1}}\|_\infty \leq \eta$, and choose π_k to be greedy with respect to $\tilde{Q}_{k-1}$. Then*

$$\|Q^* - Q_{\pi_K}\|_\infty \leq \frac{2\eta}{1-\gamma} + \frac{\gamma^K}{1-\gamma}.$$

According to Lemma 11,

$$\|Q^* - Q_{\tilde{\pi}_{K-2}}\|_\infty \leq \frac{2\eta}{1-\gamma} + \frac{\gamma^{K-2}}{1-\gamma}. \tag{18}$$

Then, since $\|Q_{\tilde{\pi}_{K-2}} - \tilde{Q}_{K-2}\|_\infty \leq \eta$, we know that

$$\|Q^* - \tilde{Q}_{K-2}\|_\infty \leq \frac{3\eta}{1-\gamma} + \frac{\gamma^{K-2}}{1-\gamma}. \tag{19}$$

The following lemma translates the gap in Q-functions to the gap in value.

Lemma 12 *(Singh and Yee, 1994) Let π be greedy with respect to a function Q. Then for any state s,*

$$V^*(s) - V_\pi(s) \leq \frac{2}{1-\gamma}\|Q^* - Q\|_\infty.$$

Since $\tilde{\pi}_{K-1}$ is greedy with respect to $\tilde{Q}_{K-2}$, we know that

$$V^*(\rho) - V_{\tilde{\pi}_{K-1}}(\rho) \leq \frac{6\eta}{(1-\gamma)^2} + \frac{2\gamma^{K-2}}{(1-\gamma)^2}. \tag{20}$$

We notice that this result is obtained by conditioning on all the previous $\ell-1$ loops and only consider the filtration of the ℓ-th loop. More specifically, given any core set $\mathcal{C}_\ell$ at the beginning of the ℓ-th loop, we have

$$\mathbb{P}\left(V^*(\rho) - V_{\tilde{\pi}_{K-1}}(\rho) \leq \frac{6\eta}{(1-\gamma)^2} + \frac{2\gamma^{K-2}}{(1-\gamma)^2} \mid \mathcal{C}_\ell\right) \geq 1 - 2KC_{\max}\exp(-2\theta^2(1-\gamma)^2 m).$$

By law of total probability we have

$$\begin{aligned}
&\mathbb{P}\left(V^*(\rho) - V_{\tilde{\pi}_{K-1}}(\rho) \leq \frac{6\eta}{(1-\gamma)^2} + \frac{2\gamma^{K-2}}{(1-\gamma)^2}\right) \\
=&\sum_{\mathcal{C}_\ell} \mathbb{P}\left(V^*(\rho) - V_{\tilde{\pi}_{K-1}}(\rho) \leq \frac{6\eta}{(1-\gamma)^2} + \frac{2\gamma^{K-2}}{(1-\gamma)^2} \mid \mathcal{C}_\ell\right) \mathbb{P}\left(\mathcal{C}_\ell\right) \\
\geq& 1 - 2KC_{\max}\exp(-2\theta^2(1-\gamma)^2 m) \sum_{\mathcal{C}_\ell} \mathbb{P}\left(\mathcal{C}_\ell\right) \\
=& 1 - 2KC_{\max}\exp(-2\theta^2(1-\gamma)^2 m).
\end{aligned}$$

With another union bound over the $C_{\max}$ loops of the virtual algorithm, we know that with probability at least

$$1 - 2KC_{\max}^2\exp(-2\theta^2(1-\gamma)^2 m), \tag{21}$$

Eq. (20) holds for all the loops. We call this event $\mathcal{E}_1$ in the following.

B.3. Analysis of the main algorithm

We now move to the analysis of the main algorithm. Throughout this section, when we mention the *final loop*, we mean the final loop of the *main algorithm*, which may not be the final loop of the virtual algorithm. We have the following result.

Lemma 13 *In the final loop of the main algorithm, all the rollout trajectories in the virtual algorithm are exactly the same as those in the main algorithm, and therefore $w_k = \tilde{w}_k$ for all $1 \leq k \leq K$.*

Proof We notice that since we only consider the final loop, in any iteration, for any state s in all the rollout trajectories in the main algorithm, and all action $a \in \mathcal{A}$, $\phi(s,a) \in \mathcal{H}$. In the first iteration, since $\pi_0 = \tilde{\pi}_0$, and the simulators are coupled, we know that all the rollout trajectories are the same between the main algorithm and the virtual algorithm, and as a result, all the Q-function estimates are the same, and thus $w_1 = \tilde{w}_1$. If we have $w_k = \tilde{w}_k$, we know that by the definition in (12), the policies π_k and $\tilde{\pi}_k$ always take the same action given s if for all $a \in \mathcal{A}$, $\phi(s,a) \in \mathcal{H}$. Again using the fact that the simulators are coupled, the rollout trajectories by π_k and $\tilde{\pi}_k$ are also the same between the main algorithm and the virtual algorithm, and thus $w_{k+1} = \tilde{w}_{k+1}$. ■

Since $\|\phi(s,a)\|_2 \leq 1$ for all s, a, we can verify that if we set $\tau \geq 1$, then after adding a state-action pair s, a to the core set, then its feature vector $\phi(s,a)$ stays in the good set $\mathcal{H}$. Recall that in the core set initialization stage of Algorithm 2, if for an action $a \in \mathcal{A}$, $\phi(\rho, a)$ is not in $\mathcal{H}$, we add ρ, a to $\mathcal{C}$. Thus, after the core set initialization stage, we have $\phi(\rho, a) \in \mathcal{H}$ for all a. Thus $\pi_{K-1}(\rho) = \tilde{\pi}_{K-1}(\rho) := a_\rho$. Moreover, according to Lemma 10, we also know that when $\mathcal{E}_1$ happens,

$$|V_{\tilde{\pi}_{K-1}}(\rho) - \tilde{w}_K^\top \phi(\rho, a_\rho)| = |Q_{\tilde{\pi}_{K-1}}(\rho, a_\rho) - \tilde{w}_K^\top \phi(\rho, a_\rho)| \leq \eta. \tag{22}$$

In the following, we bound the difference of the values of the output policy of the main algorithm π_{K-1} and the output policy of the virtual algorithm $\tilde{\pi}_{K-1}$ in the final loop of the main algorithm,

i.e., $|V_{\pi_{K-1}}(\rho) - V_{\tilde{\pi}_{K-1}}(\rho)|$. To do this, we use another auxiliary virtual policy iteration algorithm, which we call *virtual-2* in the following. Virtual-2 is similar to the virtual policy iteration algorithm in Appendix B.1. The simulator of virtual-2 is coupled with the virtual algorithm, and virtual-2 also uses the same initial policy $\hat{\pi}_0 := \pi_0$ as the main algorithm. Virtual-2 also uses Monte-Carlo rollouts with the simulator and obtains the estimated Q-function values $\hat{q}_{\mathcal{C}}$, and the linear regression coefficients are computed in the same way as (11), i.e., $\hat{w}_k = (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top \hat{q}_{\mathcal{C}}$. The virtual-2 algorithm also conducts uncertainty check in the rollout subroutine. Similar to the virtual algorithm, when it identifies an uncertain state-action pair, it records the pair and keeps running the rollout process. At the end of each loop, the virtual-2 algorithm still adds the first recorded element to the core set and discard other recorded elements. The *only difference* is that in virtual-2, we choose the virtual Q-function to be $\hat{Q}_{k-1}(s,a) := \hat{w}_k^\top \phi(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Using the same arguments in Appendix B.2, we know that with probability at least $1 - 2KC_{\max}^2 \exp(-2\theta^2(1-\gamma)^2 m)$, for all the loops and all the policy iteration steps in every loop, we have $|\hat{Q}_{k-1}(s,a) - Q_{\hat{\pi}_{k-1}}(s,a)| \le \eta$ for all (s,a) such that $\phi(s,a) \in \mathcal{H}$. We call this event $\mathcal{E}_2$. Since the simulators of virtual-2 is also coupled with that of the main algorithm, by the same argument as in Lemma 13, we know that in the last iteration of the final loop of the main algorithm, we have $\hat{\pi}_{K-1} = \pi_{K-1}$ and $\hat{w}_K = w_K$. We also know that when event $\mathcal{E}_2$ happens, in the last iteration of the all the loops of virtual-2,

$$|V_{\hat{\pi}_{K-1}}(\rho) - \hat{w}_K^\top \phi(\rho, a_\rho)| \le \eta. \tag{23}$$

Therefore, when both events $\mathcal{E}_1$ and $\mathcal{E}_2$ happen, combining (22) and (23), and using the fact that $\tilde{w}_K = w_K = \hat{w}_K$, we know that

$$\begin{aligned}
&|V_{\pi_{K-1}}(\rho) - V_{\tilde{\pi}_{K-1}}(\rho)| = |V_{\hat{\pi}_{K-1}}(\rho) - V_{\tilde{\pi}_{K-1}}(\rho)| \\
\le &|V_{\hat{\pi}_{K-1}}(\rho) - \hat{w}_K^\top \phi(\rho, a_\rho)| + |\hat{w}_K^\top \phi(\rho, a_\rho) - \tilde{w}_K^\top \phi(\rho, a_\rho)| + |\tilde{w}_K^\top \phi(\rho, a_\rho) - V_{\tilde{\pi}_{K-1}}(\rho)| \\
\le &\eta + 0 + \eta = 2\eta.
\end{aligned}$$

Combining this fact with (20) and using union bound, we know that with probability at least

$$1 - 4KC_{\max}^2 \exp(-2\theta^2(1-\gamma)^2 m), \tag{24}$$

with $C_{\max}$ defined as in (14), we have

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \le \frac{8\eta}{(1-\gamma)^2} + \frac{2\gamma^{K-2}}{(1-\gamma)^2}. \tag{25}$$

Finally, we choose the appropriate parameters. Note that we would like to ensure that the success probability in Eq. (24) is at least $1-\delta$ and at the same time, the sub-optimality (right hand side of Eq. (25)) to be as small as possible. Suppose that Assumption 3 holds, i.e, $\epsilon = 0$ in (17). It can be verified that by choosing $\tau = 1$, $\lambda = \frac{\kappa^2(1-\gamma)^4}{1024b^2}$, $n = \frac{3}{1-\gamma}\log(\frac{4(1+\log(1+\lambda^{-1})d)}{\kappa(1-\gamma)})$, $\theta = \frac{\kappa(1-\gamma)^2}{64\sqrt{d(1+\log(1+\lambda^{-1}))}}$, $K = 2 + \frac{2}{1-\gamma}\log(\frac{3}{\kappa(1-\gamma)})$, $m = 4096\frac{d(1+\log(1+\lambda^{-1}))}{\kappa^2(1-\gamma)^6}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we can ensure that the error probability is at most $1-\delta$ and $V^*(\rho) - V_{\pi_{K-1}}(\rho) \le \kappa$. Suppose that Assumption 4 holds. It can be verified that by choosing $\tau = 1$, $\lambda = \frac{\epsilon^2 d}{b^2}$, $n = \frac{1}{1-\gamma}\log(\frac{1}{\epsilon(1-\gamma)})$, $\theta = \epsilon$, $K = 2 + \frac{1}{1-\gamma}\log\left(\frac{1}{\epsilon\sqrt{d}}\right)$, $m = \frac{1}{\epsilon^2(1-\gamma)^2}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we can ensure that with probability at least $1-\delta$,

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \le \frac{74\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(1+\lambda^{-1})).$$

Appendix C. Proof of Lemma 10

To simplify notation, we write $\pi := \tilde{\pi}_{k-1}$, $\tilde{Q}(\cdot,\cdot) := \tilde{Q}_{k-1}(\cdot,\cdot)$, and $\tilde{w} = \tilde{w}_k$ in this proof. According to Eq. (16), with probability at least $1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 m)$,

$$|z_q - Q_\pi(z_s, z_a)| \le \frac{\gamma^{n+1}}{1-\gamma} + \theta$$

holds for all $z \in \mathcal{C}$. We condition on this event in the following derivation. Suppose that Assumption 4 holds. We know that there exists $w_\pi \in \mathbb{R}^d$ with $\|w_\pi\|_2 \le b$ such that for any s, a,

$$|Q_\pi(s,a) - w_\pi^\top \phi(s,a)| \le \epsilon.$$

Let $\xi := \tilde{q}_{\mathcal{C}} - \Phi_{\mathcal{C}} w_\pi$. Then we have

$$\|\xi\|_\infty \le \epsilon + \frac{\gamma^{n+1}}{1-\gamma} + \theta. \tag{26}$$

Suppose that for a state-action pair s, a, the feature vector $\phi := \phi(s,a) \in \mathcal{H}$, with $\mathcal{H}$ defined in Definition 5. Then we have

$$\begin{aligned}
|\tilde{Q}(s,a) - Q_\pi(s,a)| &\le |\phi^\top \tilde{w} - \phi^\top w_\pi| + \epsilon \\
&= |\phi^\top (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top (\Phi_{\mathcal{C}} w_\pi + \xi) - \phi^\top w_\pi| + \epsilon \\
&\le \underbrace{|\phi^\top \big(I - (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}}\big) w_\pi|}_{E_1} + \underbrace{|\phi^\top (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top \xi|}_{E_2} + \epsilon.
\end{aligned} \tag{27}$$

We then bound E_1 and E_2 in (27). Similar to Appendix A, let $\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I := V \Lambda V^\top$ be the eigendecomposition of $\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I$ with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ and V being an orthonormal matrix. Notice that for all i, $\lambda_i \ge \lambda$. Let $\alpha = V^\top \phi$. Then for E_1, we have

$$\begin{aligned}
E_1 &= |\phi^\top V \big(I - \Lambda^{-1}(\Lambda - \lambda I)\big) V^\top w_\pi| = \lambda |\phi^\top V \Lambda^{-1} V^\top w_\pi| \\
&\le \lambda b \|\alpha^\top \Lambda^{-1}\|_2 = \lambda b \sqrt{\sum_{i=1}^d \frac{\alpha_i^2}{\lambda_i^2}} \\
&\le b\sqrt{\lambda} \sqrt{\sum_{i=1}^d \frac{\alpha_i^2}{\lambda_i}},
\end{aligned} \tag{28}$$

where for the first inequality we use Cauchy-Schwarz inequality and the assumption that $\|w_{\pi_{k-1}}\|_2 \le b$, and for the second inequality we use the fact that $\lambda_i \ge \lambda$. On the other hand, since we know that $\phi \in \mathcal{H}$, we know that $\alpha^\top \Lambda^{-1} \alpha \le \tau$, i.e., $\sum_{i=1}^d \alpha_i^2 \lambda_i^{-1} \le \tau$. Combining this fact with (28), we obtain

$$E_1 \le b\sqrt{\lambda \tau}. \tag{29}$$

We now bound E_2. According to Hölder's inequality, we have

$$\begin{aligned}
E_2 &\le \|\phi^\top(\Phi_{\mathcal{C}}^\top\Phi_{\mathcal{C}}+\lambda I)^{-1}\Phi_{\mathcal{C}}^\top\|_1\|\xi\|_\infty \\
&\le \|\phi^\top(\Phi_{\mathcal{C}}^\top\Phi_{\mathcal{C}}+\lambda I)^{-1}\Phi_{\mathcal{C}}^\top\|_2\|\xi\|_\infty\sqrt{|\mathcal{C}|} \\
&= \left(\phi^\top(\Phi_{\mathcal{C}}^\top\Phi_{\mathcal{C}}+\lambda I)^{-1}\Phi_{\mathcal{C}}^\top\Phi_{\mathcal{C}}(\Phi_{\mathcal{C}}^\top\Phi_{\mathcal{C}}+\lambda I)^{-1}\phi\right)^{1/2}\|\xi\|_\infty\sqrt{|\mathcal{C}|} \\
&= \left(\alpha^\top\Lambda^{-1}(\Lambda-\lambda I)\Lambda^{-1}\alpha\right)^{1/2}\|\xi\|_\infty\sqrt{|\mathcal{C}|} \\
&= \sqrt{\sum_{i=1}^d \alpha_i^2\frac{\lambda_i-\lambda}{\lambda_i^2}}\|\xi\|_\infty\sqrt{|\mathcal{C}|} \\
&\le (\epsilon+\frac{\gamma^{n+1}}{1-\gamma}+\theta)\sqrt{\tau C_{\max}}, \qquad (30)
\end{aligned}$$

where in the last inequality we use the facts that $\sum_{i=1}^d \alpha_i^2\lambda_i^{-1}\le\tau$, Eq. (26), and Lemma 8. We can then complete the proof by combining (29) and (30).

Appendix D. Proof of Theorem 7

First, we state a general result in Szepesvári (2021) on POLITEX. Notice that in this result, we consider an arbitrary sequence of approximate Q-functions Q_k, $k=0,\ldots,K-1$, which do not have to take the form of (3).

Lemma 14 (Szepesvári (2021)) *Given an initial policy π_0 and a sequence of functions $Q_k:\mathcal{S}\times\mathcal{A}\mapsto[0,(1-\gamma)^{-1}]$, $k=0,\ldots,K-1$, construct a sequence of policies $\pi_1,\ldots,\pi_{K-1}$ according to (4) with $\alpha=(1-\gamma)\sqrt{\frac{2\log(|\mathcal{A}|)}{K}}$, then, for any $s\in\mathcal{S}$, the mixture policy $\overline{\pi}_K$ satisfies*

$$V^*(s)-V_{\overline{\pi}_K}(s)\le\frac{1}{(1-\gamma)^2}\sqrt{\frac{2\log(|\mathcal{A}|)}{K}}+\frac{2\max_{0\le k\le K-1}\|Q_k-Q_{\pi_k}\|_\infty}{1-\gamma}.$$

We then consider a virtual POLITEX algorithm. Similar to the vanilla policy iteration algorithm, in the virtual POLITEX algorithm, we begin with $\tilde{\pi}_0:=\pi_0$. In the k-th iteration, we run Monte Carlo rollout with policy $\tilde{\pi}_{k-1}$, and obtain the estimates of the Q-function values $\tilde{q}_{\mathcal{C}}$. We then compute the weight vector

$$\tilde{w}_k=(\Phi_{\mathcal{C}}^\top\Phi_{\mathcal{C}}+\lambda I)^{-1}\Phi_{\mathcal{C}}^\top\tilde{q}_{\mathcal{C}},$$

and according to Lemma 10, for any $\theta>0$, with probability at least $1-2C_{\max}\exp(-2\theta^2(1-\gamma)^2m)$, for all (s,a) such that $\phi(s,a)\in\mathcal{H}$,

$$|\tilde{w}_k^\top\phi(s,a)-Q_{\tilde{\pi}_{k-1}}(s,a)|\le b\sqrt{\lambda\tau}+\left(\epsilon+\frac{\gamma^{n+1}}{1-\gamma}+\theta\right)\sqrt{\tau C_{\max}}+\epsilon:=\eta. \qquad (31)$$

Then we define the virtual Q-function as

$$\tilde{Q}_{k-1}(s,a):=\begin{cases}\Pi_{[0,(1-\gamma)^{-1}]}(\tilde{w}_k^\top\phi(s,a)), & \phi(s,a)\in\mathcal{H},\\ Q_{\tilde{\pi}_{k-1}}(s,a), & \phi(s,a)\notin\mathcal{H},\end{cases}$$

assuming we have access to the true Q-function $Q_{\tilde{\pi}_{k-1}}(s,a)$ when $\phi(s,a) \notin \mathcal{H}$. We let the policy of the $(k+1)$-th iteration $\tilde{\pi}_k$ be

$$\tilde{\pi}_k(a|s) \propto \exp\left(\alpha \sum_{j=1}^{k-1} \tilde{Q}_{k-1}(s,a)\right). \tag{32}$$

Since we always have $Q_{\tilde{\pi}_{k-1}}(s,a) \in [0,(1-\gamma)^{-1}]$, the clipping at 0 and $(1-\gamma)^{-1}$ can only improve the accuracy of the estimation of the Q-function. Therefore, we know that with probability at least $1-2C_{\max}\exp(-2\theta^2(1-\gamma)^2 m)$, we have $\|\tilde{Q}_{k-1} - Q_{\tilde{\pi}_{k-1}}\|_\infty \le \eta$. Then, by taking a union bound over the K iterations and using the result in Lemma 14, we know that with probability at least $1-2KC_{\max}\exp(-2\theta^2(1-\gamma)^2 m)$, for any $s \in \mathcal{S}$, the virtual POLITEX algorithm satisfies

$$V^*(s) - V_{\bar{\tilde{\pi}}_K}(s) \le \frac{1}{(1-\gamma)^2}\sqrt{\frac{2\log(|\mathcal{A}|)}{K}} + \frac{2\eta}{1-\gamma}, \tag{33}$$

where $\bar{\tilde{\pi}}_K$ is the mixture policy of $\tilde{\pi}_0, \ldots, \tilde{\pi}_{K-1}$. Using another union bound over the $C_{\max}$ loops, we know that with probability at least $1-2KC_{\max}^2\exp(-2\theta^2(1-\gamma)^2 m)$, (33) holds for all the loops. We call this event $\mathcal{E}_1$ in the following.

We then consider the virtual-2 POLITEX algorithm. Similar to LSPI, the virtual-2 algorithm begins with $\hat{\pi}_0 := \pi_0$. In the k-th iteration, we run Monte Carlo rollout with policy $\hat{\pi}_{k-1}$, and obtain the estimates of the Q-function values $\hat{q}_{\mathcal{C}}$. We then compute the weight vector

$$\hat{w}_k = (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1}\Phi_{\mathcal{C}}^\top \hat{q}_{\mathcal{C}},$$

and according to Lemma 10, for any $\theta > 0$, with probability at least $1-2C_{\max}\exp(-2\theta^2(1-\gamma)^2 m)$, for all (s,a) such that $\phi(s,a) \in \mathcal{H}$,

$$|\hat{w}_k^\top \phi(s,a) - Q_{\hat{\pi}_{k-1}}(s,a)| \le \eta, \tag{34}$$

where η is defined as in (31). We also note that in the rollout process of the virtual-2 algorithm, we do not conduct the uncertainty check, i.e., we do not check whether the features are in the good set $\mathcal{H}$. By union bound, we know that with probability at least $1-2KC_{\max}^2\exp(-2\theta^2(1-\gamma)^2 m)$, (34) holds for all the K iterations of all the $C_{\max}$ loops. We call this event $\mathcal{E}_2$ in the following. In the virtual-2 algorithm, we define the approximate Q-function in the same way as the main algorithm, i.e., we define

$$\hat{Q}_{k-1}(s,a) := \Pi_{[0,(1-\gamma)^{-1}]}(\hat{w}_k^\top \phi(s,a)),$$

and we let the policy of the $(k+1)$-th iteration be

$$\hat{\pi}_k(a|s) \propto \exp\left(\alpha \sum_{j=1}^{k-1} \hat{Q}_{k-1}(s,a)\right). \tag{35}$$

We still let the simulators of all the algorithms be *coupled* in the same way described as in Appendix B.1. In addition, we also let the agent in the main algorithm be *coupled* with the virtual and virtual-2 algorithm. Take the main algorithm and the virtual algorithm as an example. Recall that in the k-th iteration of a particular loop, the main algorithm and the virtual algorithm use rollout

policies π_{k-1} and $\tilde{\pi}_{k-1}$, respectively. In the CONFIDENTROLLOUT subroutine, the agent needs to sample actions according to the policies given a state. Suppose that in the N-th time that the agent needs to take an action, the main algorithm is at state s_{main} and the virtual algorithm is at state s_{virtual}. If the two states are the same, i.e., $s_{\text{main}} = s_{\text{virtual}}$ and two distributions of actions given this state are also the same, i.e., $\pi_{k-1}(\cdot|s_{\text{main}}) = \tilde{\pi}_{k-1}(\cdot|s_{\text{virtual}})$, then the actions that the agent samples in the main algorithm and the virtual algorithm are also the same. This means that the main algorithm samples $a_{\text{main}} \sim \pi_{k-1}(\cdot|s_{\text{main}})$ and the virtual algorithm samples $a_{\text{virtual}} \sim \pi_{k-1}(\cdot|s_{\text{virtual}})$, and with probability 1, $a_{\text{main}} = a_{\text{virtual}}$. Otherwise, when $s_{\text{main}} \neq s_{\text{virtual}}$ or $\pi_{k-1}(\cdot|s_{\text{main}}) \neq \tilde{\pi}_{k-1}(\cdot|s_{\text{virtual}})$, the main algorithm and the virtual algorithm samples a new action independently. The main algorithm and the virtual-2 algorithm are coupled in the same way. We note that using the same argument as in Lemma 13, for the *final loop* of the main algorithm, all the rollout trajectories of the main, virtual, and virtual-2 algorithms are the same, which implies that $w_k = \tilde{w}_k = \hat{w}_k$ for all $1 \leq k \leq K$. This also implies that in the final loop of the main algorithm, all the policies in the K iterations are the same between the main and the virtual-2 algorithm, i.e., $\pi_k = \hat{\pi}_k$, $0 \leq k \leq K-1$. Moreover, for any state s such that $\phi(s,a) \in \mathcal{H}$ for all $a \in \mathcal{A}$, we have $\pi_k(\cdot|s) = \tilde{\pi}(\cdot|s) = \hat{\pi}_k(\cdot|s)$. Since the initial state ρ satisfies the condition that $\phi(\rho,a) \in \mathcal{H}$ for all $a \in \mathcal{A}$, we have $\pi_k(\cdot|\rho) = \tilde{\pi}(\cdot|\rho) = \hat{\pi}_k(\cdot|\rho)$.

Let $\overline{\hat{\pi}}_K$ be the policy that is uniformly chosen from $\hat{\pi}_0, \ldots, \hat{\pi}_{K-1}$ in the virtual-2 algorithm in the final loop of the main algorithm, and $\overline{\pi}_K$ be the policy that is uniformly chosen from $\pi_0, \ldots, \pi_{K-1}$ in the final loop of the main algorithm. Then we have

$$|V_{\overline{\hat{\pi}}_K}(\rho) - V_{\overline{\pi}_K}(\rho)| = \left| \frac{1}{K} \sum_{k=0}^{K-1} (V_{\hat{\pi}_k}(\rho) - V_{\pi_k}(\rho)) \right| = 0, \tag{36}$$

and when events $\mathcal{E}_1$ and $\mathcal{E}_2$ happen,

$$\begin{aligned}
&|V_{\overline{\hat{\pi}}_K}(\rho) - V_{\overline{\tilde{\pi}}_K}(\rho)| \\
\leq &\frac{1}{K} \sum_{k=0}^{K-1} |V_{\hat{\pi}_k}(\rho) - V_{\tilde{\pi}_k}(\rho)| \\
= &\frac{1}{K} \sum_{k=0}^{K-1} \left| \sum_{a \in \mathcal{A}} (\hat{\pi}_k(a|\rho) Q_{\hat{\pi}_k}(\rho,a) - \tilde{\pi}_k(a|\rho) Q_{\tilde{\pi}_k}(\rho,a)) \right| \\
\leq &\frac{1}{K} \sum_{k=0}^{K-1} \sum_{a \in \mathcal{A}} \pi_k(a|\rho) \, |Q_{\hat{\pi}_k}(\rho,a) - Q_{\tilde{\pi}_k}(\rho,a))| \\
\leq &\frac{1}{K} \sum_{k=0}^{K-1} \sum_{a \in \mathcal{A}} \pi_k(a|\rho) \left| Q_{\hat{\pi}_k}(\rho,a) - \hat{w}_k^\top \phi(\rho,a) + \hat{w}_k^\top \phi(\rho,a) - \tilde{w}_k^\top \phi(\rho,a) + \tilde{w}_k^\top \phi(\rho,a) - Q_{\tilde{\pi}_k}(\rho,a)) \right| \\
\leq &\frac{1}{K} \sum_{k=0}^{K-1} \sum_{a \in \mathcal{A}} \pi_k(a|\rho) \left(|Q_{\hat{\pi}_k}(\rho,a) - \hat{w}_k^\top \phi(\rho,a)| + |\hat{w}_k^\top \phi(\rho,a) - \tilde{w}_k^\top \phi(\rho,a)| + |\tilde{w}_k^\top \phi(\rho,a) - Q_{\tilde{\pi}_k}(\rho,a))| \right) \\
\leq &\frac{1}{K} \sum_{k=0}^{K-1} \sum_{a \in \mathcal{A}} \pi_k(a|\rho)(\eta + 0 + \eta) = 2\eta.
\end{aligned} \tag{37}$$

By combining (33), (36), and (37), and using a union bound, we obtain that with probability at least $1 - 4KC_{\max}^2 \exp(-2\theta^2(1-\gamma)^2 m)$,

$$V^*(\rho) - V_{\overline{\pi}_K}(\rho) \le \frac{1}{(1-\gamma)^2}\sqrt{\frac{2\log(|\mathcal{A}|)}{K}} + \frac{4\eta}{1-\gamma}. \tag{38}$$

Now we choose appropriate parameters to obtain the final result. When Assumption 3 holds, i.e., $\epsilon = 0$, one can verify that when we choose $\tau = 1$, $\lambda = \frac{(1-\gamma)^2\kappa^2}{256b^2}$, $K = \frac{32\log(|\mathcal{A}|)}{\kappa^2(1-\gamma)^4}$, $n = \frac{1}{1-\gamma}\log(\frac{32\sqrt{d}(1+\log(1+\lambda^{-1}))}{(1-\gamma)^2\kappa})$, $\theta = \frac{(1-\gamma)\kappa}{32\sqrt{d(1+\log(1+\lambda^{-1}))}}$, and $m = \frac{1024d(1+\log(1+\lambda^{-1}))}{(1-\gamma)^4\kappa^2}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we can ensure that with probability at least $1-\delta$, $V^*(\rho) - V_{\overline{\pi}_K}(s) \le \kappa$. When Assumption 4 holds, one can verify that when we choose $\tau = 1$, $\lambda = \frac{\epsilon^2 d}{b^2}$, $K = \frac{2\log(|\mathcal{A}|)}{\epsilon^2 d(1-\gamma)^2}$, $\theta = \epsilon$, $n = \frac{1}{1-\gamma}\log(\frac{1}{\epsilon(1-\gamma)})$, and $m = \frac{1}{\epsilon^2(1-\gamma)^2}\log(\frac{8Kd(1+\log(1+\lambda^{-1}))}{\delta})$, we can ensure that with probability at least $1-\delta$,

$$V^*(\rho) - V_{\overline{\pi}_K}(\rho) \le \frac{42\epsilon\sqrt{d}}{1-\gamma}(1+\log(1+\lambda^{-1})).$$

Appendix E. Random initial state

We have shown that with a deterministic initial state ρ, our algorithm can learn a good policy. In fact, if the initial state is random, and the agent is allowed to sample from a distribution of the initial state, denoted by ρ in this section, then we can use a simple reduction to show that our algorithm can still learn a good policy. In this case, the optimality gap is defined as the difference between the expected value of the optimal policy and the learned policy, where the expectation is taken over the initial state distribution, i.e., we hope to guarantee that $\mathbb{E}_{s\sim\rho}[V^*(s) - V_\pi(s)]$ is small.

The reduction argument works as follows. First, we add an auxiliary state s_{init} to the state space $\mathcal{S}$ and assume that the algorithm starts from s_{init}. From s_{init} and any action $a \in \mathcal{A}$, we let the distribution of the next state be $\rho \in \Delta_{\mathcal{S}}$, i.e., $P(\cdot|s_{\text{init}}, a) = \rho$. We also let $r(s_{\text{init}}, a) = 0$. Then, for any policy π, we have $\mathbb{E}_{s\sim\rho}[V_\pi(s)] = \frac{1}{\gamma}V_\pi(s_{\text{init}})$. As for the features, for any $(s,a) \in \mathcal{S}\times\mathcal{A}$, we add an extra 0 as the last dimension of the feature vector, i.e., we use $\phi^+(s,a) = [\phi(s,a)^\top\ 0]^\top \in \mathbb{R}^{d+1}$. For any $a \in \mathcal{A}$, we let $\phi^+(s_{\text{init}}, a) = [0\cdots 0\ 1]^\top \in \mathbb{R}^{d+1}$. Note that this does not affect linear realizability except a change in the upper bound on the ℓ_2 norm of the linear coefficients. Suppose that Asumption 3 holds. Suppose that in the original MDP, we have $Q_\pi(s,a) = w_\pi^\top\phi(s,a)$ with $w_\pi \in \mathbb{R}^d$. Let us define $w_\pi^+ = [w_\pi^\top\ V_\pi(s_{\text{init}})]^\top \in \mathbb{R}^{d+1}$. Then, for any $s \ne s_{\text{init}}$, we still have $Q_\pi(s,a) = (w_\pi^+)^\top\phi^+(s,a)$ since the last coordinate of $\phi^+(s,a)$ is zero. For s_{init}, we have $Q_\pi(s_{\text{init}}, a) = V_\pi(s_{\text{init}}) = (w_\pi^+)^\top\phi^+(s,a)$. The only difference is that we now have $\|w_\pi^+\|_2 \le \sqrt{b^2 + (\frac{\gamma}{1-\gamma})^2}$ since we always have $0 \le V_\pi(s_{\text{init}}) \le \frac{\gamma}{1-\gamma}$.

Then the problem reduces to the deterministic initial state case with initial state s_{init}. In the first step of the algorithm, we let $\mathcal{C} = \{(s_{\text{init}}, a, \phi^+(s_{\text{init}}, a), \mathsf{none})\}$. During the algorithm, to run rollout from any core set element z with $z_s \in \mathcal{S}$, we can use the current version of Algorithm 1. To run rollout from (s_{init}, a), we simply sample from ρ as the first "next state" and then use the simulator to keep running the following trajectory of the rollout process.