Confident Least Square Value Iteration with Local Access to a Simulator

Botao Hao[1] **Nevena Lazić**[1] **Dong Yin**[1] **Yasin Abbasi-Yadkori**[1] **Csaba Szepesvári**[1 2]
[1]DeepMind
[2]University of Alberta

Abstract

Learning with simulators is ubiquitous in modern reinforcement learning (RL). The simulator can either correspond to a simplified version of the real environment (such as a physics simulation of a robot arm) or to the environment itself (such as in games like Atari and Go). Among algorithms that are provably sample-efficient in this setting, most make the unrealistic assumption that all possible environment states are known before learning begins, or perform global optimistic planning which is computationally inefficient. In this work, we focus on simulation-based RL under a more realistic *local access* protocol, where the state space is unknown and the simulator can only be queried at states that have previously been observed (initial states and those returned by previous queries). We propose an algorithm named Confident-LSVI based on the template of least-square value iteration. Confident-LSVI incrementally builds a *coreset* of important states and uses the simulator to revisit them. Assuming that the linear function class has low approximation error under the Bellman optimality operator (a.k.a. low inherent Bellman error), we bound the algorithm performance in terms of this error, and show that it is query- and computationally-efficient.

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

1 Introduction

Modern reinforcement learning (RL) problems can be categorized into three classes based on the protocol for interaction with the environment: batch RL, online RL, and simulation-based RL. In batch RL, the agent only has access to a pre-collected dataset of experiences in the environment. In online RL, the agent can only follow the dynamics of the MDP during the learning process. In simulation-based RL, the agent has the access to a simulator and can query the simulator at particular state-action pairs. We further categorize simulation-based RL into two types:

- *Random access.* In this setting, the agent can query the simulator with *any* state-action pair of its choice to obtain a reward and a sample of the next state. This is often referred as the *access to a generative model* in literature [Kakade, 2003, Sidford et al., 2018, Yang and Wang, 2019].
- *Local access.* Here the agent can only query or revisit states that have previously been observed, such as initial states and those returned by past queries. This is a more restrictive protocol than random access, and more realistic to implement (for example, using checkpointing).

In this work, we assume the agent interacts with the environment under the local access protocol. This setting can be implemented with most simulators used in practice, but is rarely taken full advantage of. It has only recently received attention from the theory community [Yin et al., 2021, Li et al., 2021] and in empirical approaches [Ecoffet et al., 2021]. In particular, the Go-Explore algorithm [Ecoffet et al., 2021] uses state revisiting to solve a set of hard-exploration problems, including reaching the new state-of-the-art performance on Montezuma's revenge. The main message that Go-Explore delivers is that a sample-efficient agent should remember promising states that it has previously visited, and run planning or exploration from them.

Inspired by the design principles of Go-Explore, we propose a confident least-square value iteration (Confident-LSVI) algorithm that relies on the local access protocol, and advances the theoretical understanding of RL with linear function approximation. Confident-LSVI maintains a *coreset* that stores important state-action pairs. At each iteration, the agent revisits the state-action pairs in the coreset and queries the simulator with them. Whenever the agent discovers a promising new state-action pair, it updates the coreset and restarts. Our algorithm is computationally-efficient and easy to implement since we do not require any global optimistic planning which could lead to computational inefficiency [Zanette et al., 2020a, Du et al., 2021].

From a theoretical side, we assume that the linear function class has low approximation error under the Bellman optimality operator, which is often referred to as *low inherent Bellman error* (IBE) [Munos and Szepesvári, 2008, Zanette et al., 2020a,b]. We bound the policy optimization error in terms of the IBE, and show that the simulator query cost is polynomial in the feature dimension, effective horizon, and IBE. In the unrealizable case, our algorithm matches the state-of-the-art query cost bound [Lattimore et al., 2020] without the knowledge of model misspecification error, while prior works based on optimistic planning [Zanette et al., 2020a, Jin et al., 2020] require this term to correct their confidence bounds. In terms of analysis, we adapt a powerful proof technique recently proposed in Yin et al. [2021] to relate our algorithm to a virtual value iteration algorithm with uniformly bounded error under the Bellman optimality operator.

2 Related Work

The focus of our work is query-efficient learning in large MDPs with linear function approximation, assuming access to a simulator. We discuss several related works based on different structural assumptions.

Q^*-realizability In this setting, the optimal action-value function is realizable in the linear function class. For a deterministic transition kernel, Wen and Roy [2013] proposed a provably efficient algorithm with Q^*-realizability. For a stochastic transition kernel, Weisz et al. [2021b] have shown that query-efficient learning is impossible under Q^*-realizability, regardless of the algorithm or simulator interaction protocol.

Q^*-realizability with constant gap Assuming that there exists a suboptimality gap Δ_{gap} between the value of the optimal action and any suboptimal action in every state, there exist algorithms whose query cost scales polynomially in the feature dimension d, planning horizon, and $1/\Delta_{\mathrm{gap}}$ under random access [Du et al., 2020] and local access protocols [Li et al., 2021]. However, query-efficient learning with a constant gap is impossible under the online setting [Wang et al., 2021].

Low inherent Bellman error In the online RL setting, Zanette et al. [2020a] proposed an algorithm with a sublinear regret guarantee, which can be converted into a query-efficient planning algorithm through standard online-to-batch conversion [Cesa-Bianchi et al., 2004] approach. However, the algorithm of Zanette et al. [2020a] needs to solve a global optimization problem for optimistic planning, for which no efficient implementation is available. In contrast, our algorithm is computationally-efficient. Zanette et al. [2020b] developed an algorithm called FRANSIS that is both computationally efficient and query efficient but required an explorability condition.

Q_π-realizability This assumption means that the Q-functions for *all* policies can be (approximately) represented by linear function approximation in some given features. Assuming the random access to the simulator, Du et al. [2020], Lattimore et al. [2020] derived query-efficient algorithms for this setting. These algorithms assume the knowledge of full feature matrix and are not computationally-efficient. Very recently, Yin et al. [2021] developed an algorithm that is both query and computationally efficient under the local access protocol, and corresponds to a version of policy iteration.

Linear MDPs In this setting, the rewards are linear and the transition kernel has a low-rank factorization based on the features vectors. Under the random access protocol, Yang and Wang [2019] showed an optimal query-cost bound for a variance-reduction Q-learning type algorithm. Efficient learning under the linear MDP assumption has attracted much activity in recent years [Jin et al., 2020, Agarwal et al., 2020a, Wang et al., 2020, Zanette et al., 2021].

Remark 2.1. As shown in Zanette et al. [2020a], the aforementioned assumptions have the following relationship:

$$\begin{aligned}&\text{linear MDPs} \subseteq Q_\pi\text{-realizability} \subseteq Q^*\text{-realizability}\\ &\text{linear MDPs} \subseteq \text{zero IBE} \subseteq Q^*\text{-realizability}\\ &Q_\pi\text{-realizability} \neq \text{zero IBE}\,,\end{aligned}$$

where $A \subseteq B$ means assumption A is stronger than assumption B.

Du et al. [2021] introduced a more general structural framework called Bilinear Classes that can cover many of the existing popular structural assumptions. However, their algorithm still requires a global optimistic planning step which is not computationally-efficient.

Additional related works Under the linear mixture MDP assumption, Yang and Wang [2020], Cai et al. [2020], Zhou et al. [2020], Ayoub et al. [2020], Modi et al. [2020] proposed either model-based or model-free algorithms that can achieve a sub-linear regret guarantee with function approximation. Moreover, Zhou et al. [2021] proved the first minimax regret for linear mixture MDP model. Agarwal et al. [2020a] derived a $\text{poly}(d, 1/(1-\gamma), \varepsilon^{-1})$ sample cost bound for the policy cover-policy gradient algorithm. Weisz et al. [2021a] introduced TensorPlan, a query-efficient local planning under linear realizability of the optimal state-value function. Assuming the MDP has low Bellman/Witness rank, Jiang et al. [2017], Sun et al. [2019] proposed algorithms that are sample-efficient but computationally inefficient. Another line of works focus on sublinear regret guarantee with function approximation in the infinite-horizon undiscounted setting (average reward case), including POLITEX [Abbasi-Yadkori et al., 2019, Lazic et al., 2021], AAPI [Hao et al., 2021], and MDP-EXP2 [Wei et al., 2021].

3 Problem Setting

Notation We use $\Delta_\mathcal{S}$ to denote the set of probability distributions defined on any countable set $\mathcal{S}$ and write $[N] := \{1, 2, \dots, N\}$ for any positive integer N.

Infinite-horizon MDP An infinite-horizon discounted Markov decision process (MDP) can be characterized by a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}$ is the countable 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 and $\gamma \in (0, 1)$ is the discount factor. Both P and R are unknown. We let $s_0 \in \mathcal{S}$ be a fixed initial state. At each state s, if we pick action $a \in \mathcal{A}$, the environment evolves to a random next state s' according to distribution $P(s'|s, a)$ and generates a stochastic reward $r \in [0, 1]$ with $\mathbb{E}[r|s, a] = R(s, a)$.

A stationary policy $\pi : \mathcal{S} \to \Delta_\mathcal{A}$ is a mapping from a state to a distribution over actions. If for all state s, $\pi(a|s) = 1$ for some action a, we call π a *deterministic policy*. For a policy π, its value function $V_\pi(s)$ is the expected value of cumulative rewards received under policy π when starting from an arbitrary state s, i.e.,

$$V_\pi(s) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big| s_0 = s\right],$$

where $a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)$ and $\mathbb{E}^\pi$ denotes the expectation over the sample path and stochastic reward generated under policy π. The action value function $Q_\pi(s, a)$ is defined as

$$Q_\pi(s, a) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big| s_0 = s, a_0 = a\right].$$

The Bellman optimality operator for any function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as

$$\mathcal{T}Q(s, a) := R(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\max_{a' \in \mathcal{A}} Q(s', a')\right]. \tag{3.1}$$

The optimal action value function Q^* is the unique solution of Bellman optimality equation $\mathcal{T}Q = Q$. The optimal policy π^* is greedy with respect to Q^* and thus a deterministic policy.

Linear function approximation and efficient planning 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. A feature map combined with a parameter vector $w \in \mathbb{R}^d$ gives rise to the linear function $Q_w : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defined by $Q_w(s, a) = \phi(s, a)^\top w$. When the agent interacts with a large MDP, a feasible solution will approximate the optimal action value function by (linear) function approximation. Without loss of generality, we assume $\|\phi(s, a)\|_2 \le 1$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$ throughout the paper.

Let $\Pi_\gamma : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}$ be a truncation operator defined by $(\Pi_\gamma f)(s, a) = \max(\min(f(s, a), (1-\gamma)^{-1}), 0)$. We define the inherent Bellman error as follows:

Definition 3.1 (Inherent Bellman error [Munos and Szepesvári, 2008, Zanette et al., 2020a])**.** The inherent Bellman error of an MDP with a linear feature representation ϕ is denoted by $\mathcal{I}$ and defined as

$$\mathcal{I} = \sup_{v\in\mathbb{R}^d} \inf_{\substack{w\in\mathbb{R}^d \\ \|w\|_2\le b}} \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left|\phi(s,a)^\top w - \mathcal{T}\Pi_\gamma Q_v(s,a)\right| .$$

Here b is an upper bound of the ℓ_2-norm of the parameter vector w.

Remark 3.2. As shown in Zanette et al. [2020a, Proposition 3], zero inherent Bellman error assumption is a strictly weaker assumption than linear MDP assumption [Yang and Wang, 2019, Jin et al., 2020]. Intuitively, this is because in linear MDPs, the Bellman operator maps *any* function Q to a linear space, while with zero inherent Bellman error only members of the linear function space get mapped to the same space.

An efficient planning algorithm has low query and computational cost independent of the size of state space. Here, *query cost* refers to the number of queries to the simulator while *computational cost* refers to the number of logic operations. An MDP simulator is formally defined below.

Definition 3.3 (MDP simulator)**.** A simulator implementing an MDP $M = (\mathcal{S}, \mathcal{A}, R, P, \gamma)$ is a "black-box oracle" that when queried with a state action pair (s, a) returns a stochastic reward $r_{sa} \in [0, 1]$ with mean $R(s, a)$ and a random state $s'_{sa} \sim P(\cdot|s, a)$.

Local access to a simulator means the agent is only allowed to query the simulator with a state that the agent has previously observed. The goal of this work is to propose an efficient planning algorithm with local access to a simulator.

4 Algorithm

Our algorithm, named Confident-LSVI, builds on the template of least-square value iteration (LSVI) and is presented in Algorithm 1. The algorithm iteratively builds a *coreset* $\mathcal{C} \subseteq \mathcal{S} \times \mathcal{A}$ of state-action pairs whose features cover the feature directions encountered so far. During each iteration the algorithm queries the simulator with the coreset elements to obtain data for updating the Q-function estimate. If the algorithm encounters new feature directions during this process (we make this precise later), it updates the coreset and restart value iteration from the beginning. If still no new directions are encountered after K iterations, the algorithm performs a final check for the greedy policy corresponding to the last Q-function by rolling it out. If no new directions are encountered, the algorithm

Algorithm 1 Confident-LSVI

```
 1: Input: the simulator, number of iterations $K$, regularization parameter $\lambda$, uncertainty check threshold $\tau$, number of rollouts $m_1, m_2$, length of rollouts $N$
 2: Initialize: $\mathcal{C} \leftarrow \emptyset$
 3: for $a \in \mathcal{A}$ do
 4:   if $\phi(s_0,a)^\top(\Phi_\mathcal{C}^\top\Phi_\mathcal{C} + \lambda I)^{-1}\phi(s_0,a) > \tau$ then
 5:     $\mathcal{C} \leftarrow \mathcal{C} \cup \{(s_0,a)\}$
 6:   end if
 7: end for
    # start value iteration                                         (*)
 8: Initialize: $Q_0 = 0$
 9: for $k = 1,\dots,K$ do
10:   for $(s,a) \in \mathcal{C}$ do
11:     for $i = 1,\dots,m_1$ do
12:       Query the simulator with $(s,a)$, receive next state $s^i_{s,a}$ and reward $r^i_{s,a}$
13:       Run Uncertainty Check with $s^i_{s,a}, \Phi_\mathcal{C}$, parameters $\tau$, $\lambda$
14:       If not return None, restart from Line (*)
15:       Compute regression target $y_i$ as in Eq. (4.1):
          $y^i_{s,a} = r^i_{s,a} + \gamma \max_{a'} \Pi_\gamma Q_{k-1}(s^i_{s,a}, a')$
16:     end for
17:   end for
18:   Compute $w_k$ using least squares as in Eq. (4.2) and set $Q_k(s,a) = \phi(s,a)^\top w_k \ \forall s,a$
19: end for
20: Let $\pi_K$ be greedy w.r.t $Q_K$
    # final rollout check
21: for $i = 1,\dots,m_2$ do
22:   $s^i_0 \leftarrow s_0$
23:   for $j = 0,\dots,N-1$ do
24:     Query the simulator with $(s^i_j, \pi_K(s^i_j))$, receive reward $r^i_j$ and next state $s^i_{j+1}$
25:     Run Uncertainty Check with $s^i_{j+1}$, $\Phi_\mathcal{C}$, parameters $\tau$, $\lambda$
26:     If not return None, restart from Line (*)
27:   end for
28: end for
29: return the greedy policy $\pi_K$
```

returns this policy. The main benefit of local access to the simulator is that it allows us to revisit the elements of the coreset, as well as to conduct uncertainty checks on the next states and policy rollouts.

Algorithm details Let us denote $\Phi_\mathcal{C} \in \mathbb{R}^{|\mathcal{C}|\times d}$ the feature matrix of all the elements in the coreset. We write $Q_k = Q_{w_k}$ for notational simplicity and set $Q_0 =$

Algorithm 2 UNCERTAINTY CHECK

```
Input: state $s$, coreset features $\Phi_{\mathcal{C}}$, parameters $\tau$, $\lambda$
for $a \in \mathcal{A}$ do
  if $\phi(s,a)^\top(\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1}\phi(s,a) > \tau$ then
    Expand $\mathcal{C} \leftarrow \mathcal{C} \cup \{(s,a)\}$
    break
  end if
end for
return None
```

0 at the beginning of value iteration. We initialize the coreset based on the initial state s_0 and set each policy π_k to be greedy with respect to Q_k (breaking ties uniformly at random).

In each iteration, CONFIDENT-LSVI queries the simulator m_1 times at each coreset element $(s,a) \in \mathcal{C}$ to obtain samples of rewards and next states $\{r^i_{s,a}, s^i_{s,a}\}_{i=1}^{m_1}$. The algorithm checks whether the coreset should be expanded to include the new states by running the UNCERTAINTY CHECK shown in Algorithm 2 on the next states. The check determines whether there exists an action $a \in \mathcal{A}$ and next state $s^i_{s,a}$ such that

$$\phi(s^i_{s,a}, a)^\top(\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1}\phi(s^i_{s,a}, a) > \tau \,,$$

for parameters τ and λ. We say that the uncertainty check passes if it returns `None`.

- If the uncertainty check fails and returns a state action pair (s,a), the coreset $\mathcal{C}$ is expanded as $\mathcal{C} \leftarrow \mathcal{C} \cup \{(s,a)\}$, and value iteration restarts from the beginning with the updated coreset (Line $(*)$ in Algorithm 1).

- Otherwise, we compute the regression targets:

$$y^i_{s,a} = r^i_{s,a} + \gamma \max_{a'} \Pi_\gamma Q_{k-1}(s^i_{s,a}, a') \,, \tag{4.1}$$

and update the Q-function parameters using regularized least-squares:

$$Q_k(s,a) = \phi(s,a)^\top w_k \quad \forall s, a \,,$$

where

$$w_k = \operatorname*{argmin}_{w \in \mathbb{R}^d} \sum_{(s,a)\in\mathcal{C}} \frac{1}{m_1} \sum_{i=1}^{m_1} \left(y^i_{s,a} - \phi(s,a)^\top w\right)^2 + \lambda \|w\|_2^2 \,. \tag{4.2}$$

Remark 4.1. We say a *new loop* starts whenever the value iteration restarts. In each loop, the algorithm performs at most K steps of value iteration. As long as the algorithm starts a new loop, the size of the coreset $\mathcal{C}$ increases by 1.

After successfully finishing K steps of value iteration without starting a new loop, the algorithm conducts a final rollout check for the output greedy policy π_K with respect to Q_K. For each $(s,a) \in \mathcal{C}$, the algorithm runs N steps rollouts following π_K and implements the uncertainty check (Algorithm 2). If the uncertainty check fails at any point, we again expand the coreset and restart value iteration. Otherwise, we return π_K as the final policy.

Remark 4.2. In practice, rather than revisiting and querying each element of the coreset at each iteration of LSVI, we can save the data from past loops and only query the simulator at the latest addition to the coreset at the beginning of each round of LSVI.

5 Analysis

In this section, we study the query and computational cost of CONFIDENT-LSVI. We first show that the coreset expansion step can only happen a small number of times that is independent of the size of the state space.

Lemma 5.1. Under the assumption that $\|\phi(s,a)\|_2 \le 1$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, the size of the coreset is upper bounded by

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

In particular, when choosing $\tau = 1$, the size of the coreset will not exceed $4d\ln(2 + 2/\lambda)$.

This result can be derived using the technique in Russo and Van Roy [2013] as the eluder dimension of linear function class. We include a full proof in the appendix for self-completeness.

Next we state the main result.

Theorem 5.2 (Policy optimization error)**.** Let $\varepsilon > 0$ be the target accuracy. With the choice of algorithm parameters in Algorithm 1 as follows:

$$\lambda = \frac{\varepsilon^2(1-\gamma)^4}{256b^2}, \tau = 1,$$
$$m_1 = \frac{64\left(\ln(1/\delta) + d\ln(1+4\ln(2+2/\lambda)/\lambda)\right)}{\varepsilon^2(1-\gamma)^6},$$
$$m_2 = \frac{32\ln(1/\delta)}{\varepsilon^2(1-\gamma)^2},$$
$$K = \frac{\ln(8/(\varepsilon(1-\gamma)^2))}{1-\gamma}, N = \frac{\ln(8/(\varepsilon(1-\gamma)))}{1-\gamma},$$

the policy optimization error of CONFIDENT-LSVI can be upper bounded by

$$V^*(s_0) - V_{\pi_K}(s_0) \le \varepsilon + 8\sqrt{\ln\left(2 + \frac{512b^2}{d^2(1-\gamma)^4}\right)}\frac{\sqrt{d}\mathcal{I}}{(1-\gamma)^2}, \tag{5.1}$$

with probability at least $1-(K+2)C_{\max}\delta$.

In the bound of Eq. (5.1), the first term represents the target accuracy while the second term represents the model missepcification error.

Query cost Next we study the query cost of CONFIDENT-LSVI to find an ε-optimal policy and any logarithm factor is ignored. From Algorithm 1, we run at most $C_{\max}$ loops and each loop we run K steps of value iteration with m_1 times queries to the simulator at each iteration. When $\mathcal{I}=0$ in the realizable case, the total query cost to find an ε-optimal policy is at most

$$C_{\max}\left(KC_{\max}m_1 + Nm_2\right) = \widetilde{O}\left(\frac{d^3}{\varepsilon^2(1-\gamma)^7}\right).$$

When $\mathcal{I}\neq 0$, the query cost to find a $\widetilde{O}(\sqrt{d}\mathcal{I}/(1-\gamma)^2)$-optimal policy is at most

$$C_{\max}\left(KC_{\max}m_1 + Nm_2\right) = \widetilde{O}\left(\frac{d^2}{\mathcal{I}^2(1-\gamma)^3}\right).$$

Note that in our problem, the agent does not encounter a query cost when computing the feature vector $\phi(s,a)$ of a state-action pair.

Remark 5.3. Under random access protocol and linear MDPs assumption, Yang and Wang [2019] derived a $\widetilde{O}(d/\varepsilon^2(1-\gamma)^3)$ query-cost bound that matches the minimax lower bound proposed in the same paper. Since there is a large gap between our upper bound and the lower bound, it would be interesting to investigate if the more restrictive local access protocol makes the problem fundamentally harder.

Computational cost The key features of CONFIDENT-LSVI are its computational efficiency and ease of implementation. Since the value iteration steps only involve matrix multiplication and matrix inversion, the computational cost is polynomial in the aforementioned factors. Notice that the computational cost is also linear in $|\mathcal{A}|$ since in UNCERTAINTY CHECK we need to enumerate all the actions. In contrast, Zanette et al. [2020a] and Du et al. [2021] need to solve a global optimization problem whose computational tractability is unknown.

5.1 Comparison to Existing Results

We discuss our algorithm in the context of two closely related works:

- *Comparison to Yin et al. [2021].* Yin et al. [2021] studied an algorithm based on policy iteration and used the same local access protocol with Q_π-realizability. We show that the query cost of CONFIDENT-LSVI has better dependency on the effective planning horizon than Yin et al. [2021]. The main reason is that for policy iteration, we need to run rollouts of length $1/(1-\gamma)$ at each iteration while for value iteration, we just need to run rollouts of length 1. In terms of the proof technique, although we borrow the idea of introducing a virtual algorithm as a bridge, we have a different way to analyze the virtual algorithm due to the final rollout check step designed specifically to value iteration.
- *Comparison to Li et al. [2021].* Assuming a constant sub-optimality gap assumption, Li et al. [2021] also proposed a value iteration type algorithm using state revisiting but their analysis is very different from ours. They mimic the analysis of LSVI-UCB [Jin et al., 2020] in the online setting and prove the Q-function estimate is optimistic, while we mimic the analysis in the generative model setting and provide a ℓ_∞ error guarantee. In addition, their result do not allow for model misspecification error.

It is also worth emphasizing why the analysis of LSVI-UCB [Jin et al., 2020] fails for low inherent Bellman error assumption. The bonus term brings in a non-linear component that will not get mapped to the same linear function space by zero inherent Bellman error assumption but can be filtered out by the linear MDP assumption.

6 Proof Sketch

We provide a proof sketch for our main result. To analyze the algorithm, we first note that if we run approximate value iteration (AVI) with uniformly bounded error in each iteration, then using standard error propagation results, we can expect the greedy policy w.r.t. the last Q-function to perform well, as stated in the following theorem.

Theorem 6.1. Let $Q_1, Q_2, ..., Q_K$ be a sequence of Q-functions generated by running approximate value iteration. Suppose that for each $k \in [K]$ we have $||Q_k - \mathcal{T}\Pi_\gamma Q_{k-1}||_\infty \leq \eta$. Let π_k be the greedy policy with respect to Q_k. Then for any initial state s_0,

$$V^*(s_0) - V_{\pi_K}(s_0) \leq \frac{2\eta}{(1-\gamma)^2} + \frac{2\gamma^K}{(1-\gamma)^2} .$$

The detailed proof of Theorem 6.1 is deferred to Appendix A.1 and follows the steps in Munos [2005].

Unfortunately, in our algorithm, the Bellman error is only uniformly bounded on the set of state-action pairs that pass the uncertainty check (we make this precise later). Our analysis will connect the output of Confident-LSVI to the output of a value iteration algorithm that makes the same updates on this set and has uniformly bounded Bellman error outside of this set.

To make this connection, we introduce two algorithms named quasi-LSVI and virtual-LSVI. These algorithms are for analysis purposes only, and not implementable. We refer to the simulator that Confident-LSVI interacts with as the real simulator, while quasi-LSVI and virtual-LSVI interact with the quasi simulator and the virtual simulator respectively.

Step 1: Quasi-LSVI First, we note that the size of the final coreset of our real algorithm Confident-LSVI is a random variable, whose value cannot exceed $C_{\max}$, defined in Lemma 5.1. If we choose to analyze the final loop of our real algorithm, we need to condition on the fact that the policy that we output comes from the last loop; however, this type of conditioning introduces additional statistical dependencies in the analysis. The purpose of introducing quasi-LSVI, which runs exactly $C_{\max}$ loops, is to overcome this type of statistical dependencies.

- Quasi-LSVI is identical to Confident-LSVI, except that it does not restart the algorithm when the Uncertainty Check step fails. Instead, in quasi-LSVI, we record the first uncertain state-action pair that we encounter during the algorithm, and then keep running the algorithm and finish the K steps of value iteration. If the algorithm records an uncertain state-action pair, then in the next loop, we add this pair to the coreset and restart the algorithm; otherwise, we still restart the algorithm using the same coreset until we finish $C_{\max}$ loops. We will use $\bar{Q}_k$ and $\bar{w}_k$ to denote the Q-functions and corresponding weights produced by quasi-LSVI in any particular loop. Quasi-LSVI is initialized to the same function, $\bar{Q}_0 = Q_0 = 0$.
- The quasi simulator is *coupled* with the real simulator at the start of each loop. This means that when the two simulators are queried for the n^{th} time with the same state-action pair, i.e., $(s_{\text{real}}, a_{\text{real}}) = (s_{\text{quasi}}, a_{\text{quasi}})$, the next states and rewards that they return are also the same, $(s'_{\text{real}}, r_{\text{real}}) = (s'_{\text{quasi}}, r_{\text{quasi}})$. When Confident-LSVI ends the current loop, quasi-LSVI continues to finish K steps of value iteration and sample from probability transition kernel P independently from Confident-LSVI. We notice that the coupling argument and the fact that the quasi algorithm only adds the first uncertain state-action pair to the coreset ensure that before the real algorithm ends, at the beginning of each loop, the quasi-LSVI always has the same coreset as the real algorithm.

In the final loop of Confident-LSVI, since the weights produced by Confident-LSVI and quasi-LSVI, i.e., w_K and $\bar{w}_K$ are computed in the same way with exactly the same data, it is easy to see $w_K = \bar{w}_K$ in the final loop. Furthermore, both Confident-LSVI and quasi-LSVI both return the greedy policy with respect to the final Q-function. Therefore, they output exactly the same policy and thus their value functions are the same, i.e. $V_{\pi_K}(s_0) = V_{\bar{\pi}_K}(s_0)$ in the final loop of the real algorithm.

Given $\lambda, \tau > 0$, and a feature matrix $\Phi_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|\times d}$, we define $\mathcal{H} \subseteq \mathcal{S} \times \mathcal{A}$ as

$$\mathcal{H} := \left\{(s,a) : \phi(s,a)^\top(\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1}\phi(s,a) \leq \tau\right\} .$$

While we can show that our Q-function estimates have low Bellman error for state-action pairs in $\mathcal{H}$ (see Lemma 6.2 in the following), we have no control on state-action pair outside $\mathcal{H}$. To overcome this obstacle, we introduce virtual-LSVI in the next step.

Step 2: Virtual-LSVI Similarly to quasi-LSVI, virtual-LSVI runs exactly $C_{\max}$ loops and its simulator is coupled with quasi-LSVI at the start of each loop. In each loop, we run LSVI using the simulator and the current coreset, followed by the final rollout. Similarly to quasi-LSVI, virtual-LSVI records the first uncertain state-action pair that it encounters in a loop to the coreset and keeps finishing the K steps of value iteration. If an uncertain pair is recorded, virtual-LSVI adds it to the coreset and starts the next loop; and even if there is no recorded elements, it still starts the new loop with the same coreset until finishing the $C_{\max}$ loops. This design ensures that at the beginning of each loop, quasi-LSVI and virtual-LSVI have the same coreset. We use $\widetilde{Q}_k$ and $\widetilde{w}_k$ to denote the Q-functions and weight vectors produced by virtual-LSVI in a particular loop. The weight vectors are initialized to $\widetilde{w}_0 = 0$ and updated using least-squares as in quasi-LSVI. However, the *virtual Q-functions* are different and defined as follows:

$$\widetilde{Q}_k(s,a) := \begin{cases} \widetilde{w}_k^\top \phi(s,a), & (s,a) \in \mathcal{H}\,, \\ \mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a), & (s,a) \notin \mathcal{H}\,, \end{cases} \tag{6.1}$$

where the oracle knowledge of the true Bellman optimality operator $\mathcal{T}$ is assumed to be given to virtual-LSVI. The next policy $\widetilde{\pi}_k$ is defined as the greedy policy with respect to the virtual Q-function $\widetilde{Q}_k(s,a)$.

We now analyze the performance of virtual-LSVI. By the definition of $\widetilde{Q}_k$ in Eq. (6.1), the Bellman error is zero for $(s,a) \notin \mathcal{H}$. The next lemma provides a bound of Bellman error for $(s,a) \in \mathcal{H}$. Recall that per Lemma 5.1, the size of the coreset is upper-bounded by $C_{\max} = 4d\ln(2+2/\lambda)$ when $\tau = 1$.

Lemma 6.2 (Bellman error of virtual-LSVI)**.** With probability at least $1 - K\delta$, we have for any $(s,a) \in \mathcal{H}$ and any iteration $k \in [K]$,

$$\left|\widetilde{Q}_k(s,a) - \mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a)\right| \le \eta\,,$$

where

$$\eta = \sqrt{2\ln\left(\frac{1}{\delta}\right) + d\ln\left(1 + \frac{C_{\max}}{\lambda d}\right)}\frac{4\sqrt{\tau}(1-\gamma)^{-1}}{\sqrt{m_1}} + \sqrt{\lambda\tau}b + (\sqrt{C_{\max}\tau} + 1)\mathcal{I}\,,$$

and b is an upper bound of ℓ_2-norm of the parameter vector.

The proof of Lemma 6.2 is deferred to Appendix A.2. Together with a union bound over $C_{\max}$ loops, the Bellman error can be uniformly bounded: $\|\widetilde{Q}_k - \mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}\|_\infty \le \eta$ holds for any $k \in [K]$ and any loop with probability at least $1 - KC_{\max}\delta$. Combining with the finite-time error propagation analysis for AVI in Theorem 6.1, it is straightforward to have the following result.

Theorem 6.3 (Policy optimization error for virtual-LSVI)**.** Letting $\widetilde{\pi}_K$ be greedy with respect to $\widetilde{Q}_K$, we have that

$$V^*(s_0) - V_{\widetilde{\pi}_K}(s_0) \le \frac{2\eta}{(1-\gamma)^2} + \frac{2\gamma^K}{(1-\gamma)^2}\,,$$

holds for all loops of virtual-LSVI with probability at least $1 - KC_{\max}\delta$ and η is defined in Lemma 6.2.

Step 3: Analysis of quasi-LSVI The next lemma relates the rollout trajectories of virtual-LSVI and quasi-LSVI.

Lemma 6.4. When the real algorithm Confident-LSVI successfully finishes the final rollout check, we have $\widetilde{w}_K = \bar{w}_K$ and all the rollout trajectories produced by virtual-LSVI during the final rollout check are exactly the same as those produced by quasi-LSVI.

Recall that the definition of $Q_\pi(s,a)$:

$$Q_\pi(s,a) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\middle|s_0 = s, a_0 = a\right].$$

Furthermore, we define the N-step truncated Q-function:

$$Q_\pi^N(s,a) = \mathbb{E}^\pi\left[\sum_{t=0}^{N}\gamma^t r(s_t,a_t)\middle|s_0 = s, a_0 = a\right].$$

From the boundedness of reward function,

$$Q_\pi(s,a) - Q_\pi^N(s,a) \le \frac{\gamma^N}{1-\gamma}\,. \tag{6.2}$$

Note that quasi-LSVI continues to finish the K steps of value iteration and final rollouts for each loop. According to Hoeffding's inequality and using a union bound, we have with probability at least $1 - C_{\max}\delta$ for any loop

$$\left|\frac{1}{m_2}\sum_{i=1}^{m_2}\sum_{j=0}^{N-1}\gamma^j r_j^i - Q_{\bar{\pi}_K}^N(s_0, \bar{\pi}_K(s_0))\right| \le \frac{1}{1-\gamma}\sqrt{\frac{1}{2m_2}\ln\left(\frac{1}{\delta}\right)}\,,$$

where r_j^i is the reward collected by quasi-LSVI interacting with quasi simulator. Overall, we have

$$\begin{aligned} &V^*(s_0) - V_{\bar\pi_K}(s_0) = V^*(s_0) - Q_{\bar\pi_K}(s_0, \bar\pi_K(s_0)) \\ =&Q^N_{\bar\pi_K}(s_0,\bar\pi_K(s_0)) - Q_{\bar\pi_K}(s_0,\bar\pi_K(s_0)) \\ &+ \frac{1}{m_2}\sum_{i=1}^{m_2}\sum_{j=0}^{N-1}\gamma^j r_j^i - Q^N_{\bar\pi_K}(s_0,\bar\pi_K(s_0)) \\ &+ V^*(s_0) - \frac{1}{m_2}\sum_{i=1}^{m_2}\sum_{j=0}^{N-1}\gamma^j r_j^i\,. \end{aligned}$$

From Lemma 6.4, the rollout trajectectories of virtual-LSVI and quasi-LSVI are the same starting from s_0, which implies that

$$\begin{aligned} &\left|\frac{1}{m_2}\sum_{i=1}^{m_2}\sum_{j=0}^{N-1}\gamma^j r_j^i - V^*(s_0)\right| \\ \le&\frac{\gamma^N}{1-\gamma} + \frac{1}{1-\gamma}\sqrt{\frac{1}{2m_2}\ln\left(\frac{1}{\delta}\right)} \\ &\qquad + |Q_{\widetilde\pi_K}(s_0,\widetilde\pi_K(s_0)) - V^*(s_0)| \\ =&\frac{\gamma^N}{1-\gamma} + \frac{1}{1-\gamma}\sqrt{\frac{1}{2m_2}\ln\left(\frac{1}{\delta}\right)} + V^*(s_0) - V_{\widetilde\pi_K}(s_0)\,. \end{aligned}$$

Together with Theorem 6.3, we have

$$\begin{aligned} &V^*(s_0) - V_{\bar\pi_K}(s_0) \\ \le&2\left(\frac{\gamma^N}{1-\gamma} + \frac{1}{1-\gamma}\sqrt{\frac{1}{2m_2}\ln\left(\frac{1}{\delta}\right)}\right) \\ &\qquad + \frac{2\eta}{(1-\gamma)^2} + \frac{2\gamma^K}{(1-\gamma)^2}\,, \end{aligned} \tag{6.3}$$

holds with probability at least $1-(K+2)C_{\max}\delta$.

Step 4: Final query cost The final step is to simplify the policy optimization error bound and optimize the algorithm parameters.

Lemma 6.5. With the algorithm parameters chosen as in Theorem 5.2, the upper bound in Eq. (6.3) can be simplified to

$$\begin{aligned} &V^*(s_0) - V_{\bar\pi_K}(s_0) \\ &\le \varepsilon + 8\sqrt{\ln\left(2 + \frac{512b^2}{d^2(1-\gamma)^4}\right)}\frac{\sqrt{d}\mathcal{I}}{(1-\gamma)^2}\,, \end{aligned}$$

with probability at least $1-(K+2)C_{\max}\delta$.

The proof of Lemma 6.5 is deferred to Appendix A.5. From Step 1, we know that $V_{\pi_K}(s_0) = V_{\bar\pi_K}(s_0)$. This ends the proof.

7 Discussion

In this paper, we propose a query and computationally efficient local planning algorithm using local access protocol. Future work includes extending this idea to richer function classes, such as Bilinear Classes [Du et al., 2021]. Note that in our paper, the feature mapping is given. Thus it is also very interesting to study the representation learning [Agarwal et al., 2020b, Modi et al., 2021] under local access protocol.

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, Akshay Krishnamurthy, and Wen Sun. FLAMBE: Structural complexity and representation learning of low rank MDPs. *arXiv preprint arXiv:2006.10814v2*, 2020b.

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.

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

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. First return, then explore. *Nature*, 590(7847):580–586, 2021.

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.

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

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

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, 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.

Aditya Modi, Nan Jiang, Ambuj Tewari, and Satinder Singh. Sample complexity of reinforcement learning using linearly combined model ensembles. In *International Conference on Artificial Intelligence and Statistics*, pages 2010–2020. PMLR, 2020.

Aditya Modi, Jinglin Chen, Akshay Krishnamurthy, Nan Jiang, and Alekh Agarwal. Model-free representation learning and exploration in low-rank mdps. *arXiv preprint arXiv:2102.07035*, 2021.

Rémi Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1006. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.

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.

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.

Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pages 2898–2933, 2019.

Ruosong Wang, Simon S Du, Lin Yang, and Russ R Salakhutdinov. On reward-free reinforcement learning with linear function approximation. *Advances in neural information processing systems*, 33:17816–17826, 2020.

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. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 3021–3029, 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.

Dong Yin, Botao Hao, Yasin Abbasi-Yadkori, Nevena Lazić, and Csaba Szepesvári. Efficient local planning with linear function approximation. *arXiv preprint arXiv:2108.05533*, 2021.

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, 2020a.

Andrea Zanette, Alessandro Lazaric, Mykel J Kochenderfer, and Emma Brunskill. Provably efficient reward-agnostic navigation with linear value iteration. *Advances in Neural Information Processing Systems*, 33:11756–11766, 2020b.

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.

Dongruo Zhou, Quanquan Gu, and Csaba Szepesvari. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*, pages 4532–4576. PMLR, 2021.

A Proofs

A.1 Proof of Theorem 6.1

Proof. We first write

$$\begin{aligned}||Q_k - Q_*||_\infty &\le ||Q_k - \mathcal{T}\Pi_\gamma Q_{k-1}||_\infty + ||\mathcal{T}\Pi_\gamma Q_{k-1} - \mathcal{T}Q_*||_\infty \\ &\le ||Q_k - \mathcal{T}\Pi_\gamma Q_{k-1}||_\infty + \gamma||\Pi_\gamma Q_{k-1} - Q_*||_\infty \\ &\le \eta + \gamma||Q_{k-1} - Q_*||_\infty\,,\end{aligned}$$

where the second inequality follows from the contraction of the Bellman operator and the third inequality follows from the error assumption and the fact that Π_γ can only make error smaller. Thus expanding the error expression and using the fact that Q_* in the valid range $\left[0,(1-\gamma)^{-1}\right]$, we get the result

$$||Q_K - Q_*||_\infty \le \sum_{k=0}^{K-1}\gamma^k\eta + \gamma^K||Q_0 - Q_*||_\infty \le \frac{\eta}{1-\gamma} + \frac{\gamma^K}{1-\gamma}\,.$$

Using the standard result in Singh and Yee [1994], we have

$$V^*(s_0) - V_{\pi_K}(s_0) \le \frac{2}{1-\gamma}\|Q_K - Q_*\|_\infty \le \frac{2\eta}{(1-\gamma)^2} + \frac{2\gamma^K}{(1-\gamma)^2}\,.$$

□

A.2 Proof of Lemma 6.2

From Definition 3.1, there exists some $v_k \in \mathbb{R}^d$ with $\|v_k\|_2 \le b$ such that for any (s,a),

$$\left|\mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a) - \phi(s,a)^\top v_k\right| \le \mathcal{I}\,. \tag{A.1}$$

For any $(s,a)\in\mathcal{H}$, we use the definition of $\widetilde{Q}_k$ in Eq. (6.1) such that

$$\begin{aligned}&\left|\widetilde{Q}_k(s,a) - \mathcal{T}\Pi_\gamma\widetilde{Q}_{k-1}(s,a)\right| \\ =&\left|\phi(s,a)^\top\widetilde{w}_k - \phi(s,a)^\top v_k + \phi(s,a)^\top v_k - \mathcal{T}\Pi_\gamma\widetilde{Q}_{k-1}(s,a)\right| \\ \le&\left|\phi(s,a)^\top\left(\widetilde{w}_k - v_k\right)\right| + \mathcal{I}\,,\end{aligned} \tag{A.2}$$

where $\widetilde{w}_k$ is the regularized least square estimator. Let's denote $V(\lambda) = \sum_{(s,a)\in\mathcal{C}}\phi(s,a)\phi(s,a)^\top + \lambda I$. We decompose the main term into the errors due to the regularization, noise and bias respectively as follows:

$$\begin{aligned}\phi(s,a)^\top\left(\widetilde{w}_k - v_k\right) &= \phi(s,a)^\top\left(V(\lambda)^{-1}\sum_{(s,a)\in\mathcal{C}}\frac{1}{m_1}\sum_{i=1}^{m_1}\left(r^i_{s,a} + \gamma\max_a\Pi_\gamma\widetilde{Q}_{k-1}(s^i_{s,a},a)\right) - v_k\right) \\ &= \underbrace{-\lambda\phi(s,a)^\top V(\lambda)^{-1}v_k}_{I_1} \\ &\quad\underbrace{+\,\phi(s,a)^\top V(\lambda)^{-1}\sum_{(s,a)\in\mathcal{C}}\phi(s,a)\left(\frac{1}{m_1}\sum_{i=1}^{m_1}\left(r^i_{s,a} + \gamma\max_a\Pi_\gamma\widetilde{Q}_{k-1}(s^i_{s,a},a) - \mathcal{T}\Pi_\gamma\widetilde{Q}_{k-1}(s,a)\right)\right)}_{I_2} \\ &\quad\underbrace{+\,\phi(s,a)^\top V(\lambda)^{-1}\sum_{(s,a)\in\mathcal{C}}\phi(s,a)\left(\mathcal{T}\Pi_\gamma\widetilde{Q}_{k-1}(s,a) - \phi(s,a)^\top v_k\right)}_{I_3}\,.\end{aligned}$$

For a vector $x\in\mathbb{R}^d$ and a matrix $A\in\mathbb{R}^{d\times d}$, we define $\|x\|_A = \sqrt{x^\top Ax}$. According to the definition of good set $\mathcal{H}$ in Eq. (6), we have $\|\phi(s,a)\|^2_{V(\lambda)^{-1}} \le \tau$ for $(s,a)\in\mathcal{H}$.

- *Bounding I_1.* By Cauchy–Schwarz inequality,
$$|I_1| \le \lambda \|\phi(s,a)\|_{V(\lambda)^{-1}} \|v_k\|_{V(\lambda)^{-1}}\,.$$
Since $\lambda_{\min}(V(\lambda)) \ge \lambda$, we have $v_k^\top V(\lambda)^{-1} v_k \le \|v_k\|_2^2/\lambda \le b^2/\lambda$. This implies
$$|I_1| \le \sqrt{\lambda \tau} b\,. \tag{A.3}$$

- *Bounding I_2.* From the definition of Bellman optimality operator, the true regression target is
$$\mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_a \Pi_\gamma \widetilde{Q}_{k-1}(s',a)\right]\,.$$
Note that $r_{s,a}^i + \gamma \max_a \Pi_\gamma \widetilde{Q}_{k-1}(s_{s,a}^i, a)$ is an unbiased estimate of $\mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a)$ and is bounded by $[0, 1+(1-\gamma)^{-1}]$. Let
$$Z_i = r_{s,a}^i + \gamma \max_a \Pi_\gamma \widetilde{Q}_{k-1}(s_{s,a}^i, a) - \mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a).$$
Then we have $Z_i/(\sqrt{m_1}(1+(1-\gamma)^{-1}))$ is bounded by $1/\sqrt{m_1}$. By the rotation invariance of sub-Gaussian random variable, the sub-Gaussian norm of $\sum_{i=1}^{m_1} Z_i/(\sqrt{m_1}(1+(1-\gamma)^{-1}))$ is bounded by 1. Then we can rewrite
$$\begin{aligned}
&\left\|\sum_{(s,a)\in\mathcal{C}} \phi(s,a)\left(\frac{1}{m_1}\sum_{i=1}^{m_1}\left(r_{s,a}^i + \gamma \max_a \Pi_\gamma \widetilde{Q}_{k-1}(s_{s,a}^i,a)\right) - \mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a)\right)\right\|_{V(\lambda)^{-1}} \\
=&\left\|\frac{1+(1-\gamma)^{-1}}{\sqrt{m_1}}\sum_{(s,a)\in\mathcal{C}}\phi(s,a)\sum_{i=1}^{m_1}\frac{Z_i}{\sqrt{m_1}(1+(1-\gamma)^{-1})}\right\|_{V(\lambda)^{-1}} \\
=&\frac{1+(1-\gamma)^{-1}}{\sqrt{m_1}}\left\|\sum_{(s,a)\in\mathcal{C}}\phi(s,a)\sum_{i=1}^{m_1}\frac{Z_i}{\sqrt{m_1}(1+(1-\gamma)^{-1})}\right\|_{V(\lambda)^{-1}}\,.
\end{aligned}$$
Using the self-normalized martingale inequality (Theorem 20.4 in Lattimore and Szepesvári [2020]), the following holds with probability at least $1-\delta$
$$\left\|\sum_{(s,a)\in\mathcal{C}}\phi(s,a)\sum_{i=1}^{m_1}\frac{Z_i}{\sqrt{m_1}(1+(1-\gamma)^{-1})}\right\|_{V(\lambda)^{-1}} \le \sqrt{2\ln\left(\frac{1}{\delta}\right) + \ln\left(\frac{\det(V(\lambda))}{\lambda^d}\right)}\,.$$
Since we have assumed $\|\phi(s,a)\|_2 \le 1$,
$$\ln\left(\frac{\det(V(\lambda))}{\lambda^d}\right) \le \ln\left[\left(1+\frac{|\mathcal{C}|}{\lambda d}\right)^d\right] \le d\ln\left(1+\frac{|\mathcal{C}|}{\lambda d}\right)\,.$$
Therefore, we have with probability at least $1-\delta$,
$$\begin{aligned}
I_2 &\le \|\phi(s,a)\|_{V(\lambda)^{-1}}\left\|\sum_{(s,a)\in\mathcal{C}}\phi(s,a)\left(\frac{1}{m_1}\sum_{i=1}^{m_1}\left(r_{s,a}^i + \gamma\max_a \Pi_\gamma \widetilde{Q}_{k-1}(s_{s,a}^i,a)\right) - \mathcal{T}\Pi_\gamma\widetilde{Q}_{k-1}(s,a)\right)\right\|_{V(\lambda)^{-1}} \\
&\le \sqrt{\tau}\sqrt{2\ln\left(\frac{1}{\delta}\right) + d\ln\left(1+\frac{|\mathcal{C}|}{\lambda d}\right)}\frac{2(1+(1-\gamma)^{-1})}{\sqrt{m_1}}\,.
\end{aligned} \tag{A.4}$$

- *Bounding I_3.* According to Lemma B.1 and Eq. (A.1),
$$\begin{aligned}
|I_3| &\le \|\phi(s,a)\|_{V(\lambda)^{-1}}\left\|\sum_{(s,a)\in\mathcal{C}}\phi(s,a)\left(\mathcal{T}\Pi_\gamma\widetilde{Q}_{k-1}(s,a) - \phi(s,a)^\top v_k\right)\right\|_{V(\lambda)^{-1}} \qquad (A.5)\\
&\le \sqrt{\tau}\sqrt{|\mathcal{C}|\mathcal{I}^2}\,.
\end{aligned}$$

Overall, putting Eqs (A.2)-(A.5) together, we have

$$\begin{aligned}&\left|\widetilde{Q}_k(s,a) - \mathcal{T}\Pi_\gamma \widetilde{Q}_{k-1}(s,a)\right| \\ &\leq \left(\sqrt{\lambda}b + \sqrt{2\ln\left(\frac{1}{\delta}\right) + d\ln\left(1+\frac{|\mathcal{C}|}{\lambda d}\right)}\frac{4(1-\gamma)^{-1}}{\sqrt{m_1}}\right)\sqrt{\tau} + (\sqrt{|\mathcal{C}|\tau}+1)\mathcal{I}\,,\end{aligned}$$

with probability at least $1-\delta$. Taking a union bound over K iterations, we finish the proof.

A.3 Proof of Lemma 6.4

When the algorithm successfully finishes the final rollout check, both virtual-LSVI and quasi-LSVI are in the last loop. Then in any iteration and any rollout trajectories of quasi-LSVI, all the state-action pairs belong to $\mathcal{H}$. In the first iteration, since $\bar{\pi}_0 = \widetilde{\pi}_0$ and two simulators are coupled, two regression targets are the same and thus $\bar{w}_1 = \widetilde{w}_1$. If we have $\bar{w}_k = \widetilde{w}_k$, by the definition of virtual Q-function in Eq. (6.1), $\bar{\pi}_k$ and $\widetilde{\pi}_k$ always take the same action given s when for all $a \in \mathcal{A}$, $(s,a) \in \mathcal{H}$. Again using the fact that the simulators are coupled, the rollout trajectories by π_k and $\widetilde{\pi}_k$ are also the same between the main algorithm and the virtual algorithm, and thus $\bar{w}_{k+1} = \widetilde{w}_{k+1}$.

A.4 Proof of Lemma 5.1

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 $\phi_t \in \mathbb{R}^d$, $\|\phi_t\|_2 \leq 1$, such that

$$\phi_t^\top(\Phi_{t-1}^\top\Phi_{t-1} + \lambda I)^{-1}\phi_t > \tau\,, \tag{A.6}$$

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

$$\begin{aligned}\det(\Sigma_t) &= (1+\phi_t^\top\Sigma_{t-1}^{-1}\phi_t)\det(\Sigma_{t-1}) > (1+\tau)\det(\Sigma_{t-1}) \\ &> \cdots > (1+\tau)^t\det(\Sigma_0) = (1+\tau)^t\lambda^d\,,\end{aligned} \tag{A.7}$$

where the inequality is due to (A.6). Since $\det(\Sigma_t)$ is the product of all the eigenvalues of Σ_t, according to the AM-GM inequality, we have

$$\det(\Sigma_t) \leq \left(\frac{\operatorname{tr}(\Sigma_t)}{d}\right)^d = \left(\frac{\operatorname{tr}(\sum_{i=1}^t \phi_i\phi_i^\top) + \operatorname{tr}(\lambda I)}{d}\right)^d \leq \left(\frac{t}{d}+\lambda\right)^d\,, \tag{A.8}$$

where in the second inequality we use the fact that $\|\phi_i\|_2 \leq 1$. Combining (A.7) and (A.8), we know that t must satisfy

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

which is equivalent to

$$(1+\tau)^{\frac{t}{d}} < \frac{t}{\lambda d} + 1. \tag{A.9}$$

We note that if $t \leq d$, the result of the size of the core set in Lemma 5.1 automatically holds. Thus, we only consider the situation here $t > d$. In this case, the condition (A.9) implies

$$\begin{aligned}\frac{t}{d}\ln(1+\tau) < \ln(1+\frac{t}{\lambda d}) < \ln(\frac{t}{d}(1+\frac{1}{\lambda})) &= \ln(\frac{t}{d}) + \ln(1+\frac{1}{\lambda}) \\ &= \ln\left(\frac{t\tau}{d(1+\tau)}\right) + \ln(\frac{1+\tau}{\tau}) + \ln(1+\frac{1}{\lambda})\,.\end{aligned} \tag{A.10}$$

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

$$\frac{t\tau}{d(1+\tau)} < \frac{t\tau}{ed(1+\tau)} + \ln(\frac{1+\tau}{\tau}) + \ln(1+\frac{1}{\lambda}), \tag{A.11}$$

which implies

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

This ends the proof.

A.5 Proof of Lemma 6.5

By elementary change of base formula and Taylor expansion, we have

$$\ln_{1/\gamma}(x) = \frac{\ln(x)}{\ln(1/\gamma)} \approx \frac{\ln(x)}{1-\gamma}.$$

Let $\varepsilon > 0$ be the target accuracy. When we choose the number of iterations $K = \ln(8/(\varepsilon(1-\gamma)^2))/(1-\gamma)$, we have

$$\frac{2\gamma^K}{(1-\gamma)^2} \le \frac{1}{4}\varepsilon. \tag{A.12}$$

When we choose the length of rollouts $N = \ln(8/(\varepsilon(1-\gamma)))/(1-\gamma)$, we have

$$\frac{2\gamma^K}{(1-\gamma)} \le \frac{1}{4}\varepsilon. \tag{A.13}$$

When we choose the number of rollouts $m_2 = 32\ln(1/\delta)/((1-\gamma)^2\varepsilon^2)$, we have

$$\frac{2}{1-\gamma}\sqrt{\frac{1}{2m_2}\ln\left(\frac{1}{\delta}\right)} \le \frac{1}{4}\varepsilon. \tag{A.14}$$

Setting the uncertainty check threshold $\tau = 1$, the Q-function estimation error reduces to

$$\eta = \sqrt{2\ln\left(\frac{1}{\delta}\right) + d\ln\left(1+\frac{C_{\max}}{\lambda d}\right)}\frac{4(1-\gamma)^{-1}}{\sqrt{m_1}} + \sqrt{\lambda}b + (\sqrt{C_{\max}}+1)\mathcal{I},$$

and the maximum size of the coreset $C_{\max} = 4d\ln(2+2/\lambda)$. Letting the number of queries during value iteration step

$$m_1 = \frac{1028\left(\ln(1/\delta) + d\ln\left(1+\frac{C_{\max}}{\lambda d}\right)\right)}{\varepsilon^2(1-\gamma)^6},$$

and the regularization parameter

$$\lambda = \frac{\varepsilon^2(1-\gamma)^4}{256b^2},$$

we have

$$\frac{2\eta}{(1-\gamma)^2} \le \frac{1}{4}\varepsilon + \frac{4\sqrt{C_{\max}}}{(1-\gamma)^2}\mathcal{I}. \tag{A.15}$$

Putting Eqs. (A.12)-(A.15) together, we have

$$|V_{\bar{\pi}_K}(s_0) - V^*(s_0)| \le \varepsilon + \frac{4\sqrt{C_{\max}}}{(1-\gamma)^2}\mathcal{I} = \varepsilon + 8\sqrt{\ln\left(2+\frac{512b^2}{d^2(1-\gamma)^4}\right)}\frac{\sqrt{d}\mathcal{I}}{(1-\gamma)^2},$$

with probability at least $1-(K+2)C_{\max}\delta$. This ends the proof.

B Auxiliary Lemmas

Lemma B.1 (Lemma 8 in Zanette et al. [2020a])**.** Let $\{a_i\}_{i=1,\dots,n}$ be any sequence of vectors in $\mathbb{R}^d$ and $\{b_i\}_{i=1,\dots,n}$ be any sequence of scalars that $|b_i| \leq \epsilon$. For any $\lambda \geq 0$ we have

$$\left\| \sum_{i=1}^{n} a_i b_i \right\|^2_{(\sum_{i=1}^{n} a_i a_i^\top + \lambda I)^{-1}} \leq n\epsilon^2 \,.$$