Confident Approximate Policy Iteration for Efficient Local Planning in q^π-realizable MDPs

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider approximate dynamic programming in γ-discounted Markov Decision Processes and apply it to approximate planning with linear value function approximation. Our first contribution is a new variant of Approximate Policy Iteration (API), which we call Confident Approximate Policy Iteration (CAPI). CAPI is shown to produce a deterministic stationary policy with an optimal error bound scaling linearly with product of the effective horizon H and the worst-case approximation error ε of the action-value functions q^π of stationary policies. This improvement over API (whose error scales with H^2) comes at the price of an H-fold increase in memory cost. Unlike Scherrer and Lesner [2012], who recommended returning a non-stationary policy to achieve a similar improvement (with the same memory overhead), we are able to stick to stationary policies. This allows for our second contribution, which is the application of CAPI to planning with local access to a simulator and d-dimensional linear function approximation. As such, we design a planning algorithm that applies CAPI to obtain a sequence of policies with successively refined accuracies on a dynamically evolving set of states. The algorithm outputs an $O(\sqrt{d}H\varepsilon)$-optimal policy after issuing $\tilde{O}(dH^4/\varepsilon^2)$ queries to the simulator, simultaneously achieving the best known bounds for both accuracy and query complexity, while earlier algorithms in the literature achieve only one of them. These improvements come at the expense of a mild (polynomial) increase in memory and computational costs of both the algorithm and the policy found by the algorithm.

1 Introduction

Finding near-optimal policies in Markov Decision Processes (MDPs) using linear function approximation to approximate value functions lies at the heart of Reinforcement Learning. A flurry of recent results aims at solving this problem efficiently with varying models of interaction with the MDP. In this paper we focus on the problem of *planning with a simulator*. A simulator is an oracle that, given a state-action pair as a query, returns a transition and reward generated from the same distribution as the transition and reward kernels of the MDP it simulates. Depending on the application, such a simulator is often readily available (e.g., in chess, go, Atari). Planning with simulator access comes with great benefits: for example, in a recent work, Wang et al. [2021] showed that a near-optimal policy can be found exponentially more efficiently in some settings when a simulator of the MDP is available, compared to the online model where the learner has to interact sequentially with the environment without using a simulator.

Our setting of *offline planning* considers the problem of finding a policy with near-optimal value at a given start state s_0 in the MDP. The planner can issue queries to the simulator, and has to find and output a near-optimal policy with high probability. The efficiency of a planner that achieves a given

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

target suboptimality is measured in three ways: *query cost*, which is the number of queries issued to the simulator, *computational cost*, which is the number of operations used, and *memory cost*, which is the amount of memory used. We adopt the real computation model for these costs. There are several interaction models between the planner and the simulator [Yin et al., 2022]. The most permissive one is called the *generative model*, or *random access*. Here, the planner receives the set of all states and is allowed to issue queries for any state and action. Coding a simulator that supports this model can be challenging, as oftentimes the set of states is computationally difficult to describe. Instead of *random access*, we consider the more challenging *local access* setting, where the planner only sees the starting state and the set of states received as a result to a query to the simulator. Consequently, the queries issued have to be for a state that has already been encountered this way (and any available action), while the simulator needs to support the ability to reset the MDP state *only* to previously seen states. A simple approach to support this model is saving and reloading checkpoints during the operation of the simulator.

To handle large, possibly infinite state spaces, we use linear function approximation used to approximate the action-value functions q^π of stationary, deterministic policies (for background on MDPs, see the next section):

Definition 1.1 (q^π-realizability: uniform policy value-function approximation error)**.** *The uniform policy value-function approximation error induced by a feature map φ mapping state-action pairs to the Euclidean ball of radius L centered at zero in $\mathbb{R}^d$ is $\varepsilon = \sup_\pi \inf_{\theta:\|\theta\|_2 \le L} \sup_{(s,a)} |q^\pi(s,a) - \langle \varphi(s,a), \theta \rangle|$, where the outermost supremum is over all possible stationary deterministic policies.*

Note that the features and the MDP together induce an approximation error of a certain size. The goal is to design algorithms that scale gracefully with the uniform approximation error at the expense of controlled computation cost. To achieve nontrivial guarantees, the uniform approximation error needs to be "small". This (implicit) assumption is stronger than the $q^\star$-realizable assumption, which Weisz et al. [2021] showed an exponential query complexity lower bound for. At the same time, it is (strictly) weaker than the linear MDP assumption [Zanette et al., 2020], for which there are efficient algorithms that do not require a simulator [Jin et al., 2020], even in the reward-free setting [Wagenmaker et al., 2022].

In the *local access* setting, the features $\varphi(s,a)$ are revealed to the planner *only* for state-action pairs (s,a) where the state s has already been encountered. In contrast, in the *random access* setting, the whole feature map $\varphi(\cdot,\cdot)$, of (possibly infinite) size $|\mathcal{S}||\mathcal{A}|d$, is given to the planner as input. In the latter setting, when only the query cost is counted, Du et al. [2019], Lattimore et al. [2020] proposed algorithms (the latter working in the misspecified, $\varepsilon > 0$ regime) that both issue a number of queries that is polynomial in the relevant parameters, but require a barycentric spanner or near-optimal design of the input features. In the worst case, computing this set scales polynomially in $|\mathcal{S}|$ and $|\mathcal{A}|$, which can be prohibitive.

Yin et al. [2022] propose two algorithms for the *local access* model that we also consider. The first, CONFIDENT MC-LSPI, relies on policy iteration from a "core-set" of informative state-action pairs, similarly to the algorithm presented in this work. The second, CONFIDENT MC-POLITEX is its regularized version, based on the work of Abbasi-Yadkori et al. [2019], that achieves a better policy suboptimality bound at the expense of a higher query cost. Our algorithm, CAPI-QPI-PLAN (given Algorithm 3), achieves the best of all worlds by achieving the suboptimality bound of Yin et al. [2022] with the query cost of Lattimore et al. [2020] in the *local access* setting (see Table 1 for a comparison).

Theorem 1.2. *For any $\delta \in (0,1]$, $\omega > 0$, $s_0 \in \mathcal{S}$, with probability at least $1-\delta$, Algorithm 3 (called with these parameters) returns a policy π with*

$$v^\star(s_0) - v^\pi(s_0) = \tilde{O}\left((\varepsilon + \omega)\sqrt{d}(1-\gamma)^{-1}\right), \tag{1}$$

while executing at most $\tilde{O}\left(d(1-\gamma)^{-4}\omega^{-2}\right)$ queries, where ε is as defined in Definition 1.1.

This result comes at the expense of increased memory and computational costs (Theorem 1.3): compared to CONFIDENT MC-LSPI, the algorithm's memory and computational costs increase by a factor of the effective horizon $H = \tilde{O}(1/(1-\gamma))$, and the policy outputted by the algorithm uses a dH factor more memory for storage and a d^2H factor more computation to evaluate.

Table 1: Comparison of suboptimality and query complexity guarantees of various planners. Drawbacks are highlighted with red.

Algorithm [Publication]	Query cost	Suboptimality	Access model
MC-LSPI [Lattimore et al., 2020]	$\tilde{O}(\frac{d}{\varepsilon^2(1-\gamma)^4})$	$\tilde{O}(\frac{\varepsilon\sqrt{d}}{(1-\gamma)^2})$	random access
CONFIDENT MC-LSPI [Yin et al., 2022]	$\tilde{O}(\frac{d^2}{\varepsilon^2(1-\gamma)^4})$	$\tilde{O}(\frac{\varepsilon\sqrt{d}}{(1-\gamma)^2})$	local access
CONFIDENT MC-POLITEX [Yin et al., 2022]	$\tilde{O}(\frac{d}{\varepsilon^4(1-\gamma)^5})$	$\tilde{O}(\frac{\varepsilon\sqrt{d}}{1-\gamma})$	local access
CAPI-QPI-PLAN [This work]	$\tilde{O}(\frac{d}{\varepsilon^2(1-\gamma)^4})$	$\tilde{O}(\frac{\varepsilon\sqrt{d}}{1-\gamma})$	local access

Theorem 1.3 (Memory and compute cost)**.** *The memory and computational cost of running Algorithm 3 are*

$$\tilde{O}\left(d^2/(1-\gamma)\right) \quad \text{and} \quad \tilde{O}\left(d^4|\mathcal{A}|(1-\gamma)^{-5}\omega^{-2}\right),$$

respectively, while the memory and computational costs of storing and evaluating the final policy outputted by Algorithm 3 (respectively) are

$$\tilde{O}\left(d^2/(1-\gamma)\right) \quad \text{and} \quad \tilde{O}\left(d^3|\mathcal{A}|/(1-\gamma)\right).$$

At the heart of the new result is a new algorithm, which we call confident approximate policy iteration (CAPI). This algorithm belongs to the family of approximate dynamic programming algorithms [Bertsekas, 2012, Munos, 2003, 2005]. As described by Scherrer and Lesner [2012] (see also [Russo, 2020]), the "classical" approximate dynamic programming algorithms (approximate policy and value iteration) can inflate the value function evaluation error by a factor of H^2 where $H = 1/(1-\gamma)$. Scherrer and Lesner [2012] proposed to keep and use all policies obtained while running any of approximate value or policy iteration, in the order in which they were produced, to reduce this inflation factor from H^2 to H. Since the policies to be evaluated are nonstationary themselves, including these policies may increase the error ε as compared to that given in Definition 1.1, which is problematic. Another alternative would be to start from Confident MC-Politex: While it is possible to show that with this one does not need to change the definition of the error (this requires an argument as Politex uses stochastic policies), Politex produces a sequence of policies that converges not a geometric but only at a polynomial rate and thus without much further work, it will be hard to match the sample and computational cost of CAPI with it. As such, we leave the investigation of whether this is possible for future work.

The rest of the paper is organized as follows: The model and notation are introduced in Section 2. Our algorithm, CAPI-QPI-PLAN, is based on a novel variant of Approximate Policy Iteration (API), called Confident Approximate Policy Iteration (CAPI), which produces a deterministic stationary policy with an optimal error bound (scaling linearly with the effective horizon H), improving upon API, whose error scales quadratically with H. CAPI is introduced and analyzed in Section 3. Planning with q^π realizability is introduced in Section 4, with CAPI-QPI-PLAN being built-up and analyzed in Sections 4.1 and 4.2. In particular, the proof of Theorem 1.2 is given in Section 4.2. Several proofs are relegated to appendices, in particular, Theorem 1.3 is proved and implementation details of CAPI-QPI-PLAN are discussed in Appendix F.

2 Notation and Preliminaries

Let $\mathbb{N} = \{0, 1, \ldots\}$ denote the set of natural numbers, $\mathbb{N}^+ = \{1, \ldots\}$ the positive integers. For some integer i, let $[i] = \{0, \ldots, i-1\}$. For $x \in \mathbb{R}$, let $\lceil x \rceil$ denote the smallest integer i such that $i \geq x$. For a positive definite $V \in \mathbb{R}^{d\times d}$ and $x \in \mathbb{R}^d$, let $\|x\|_V^2 = x^\top V x$. For matrices A and B, we say that $A \geq B$ if $A - B$ is positive semidefinite. Let $\mathbb{I}$ be the d-dimensional identity matrix. For compatible vectors x, y, let $\langle x, y\rangle$ be their inner product: $\langle x, y\rangle = x^\top y$. Let $\mathcal{M}_1(X)$ denote the space of probability distributions supported on the set X. Throughout, we assume that the σ-algebra is implicit. For a distribution D, let $\mathrm{supp}(D)$ denote its support. We write $a \approx_\varepsilon b$ for $a, b, \varepsilon \in \mathbb{R}$ if $|a - b| \leq \varepsilon$. We denote by $\tilde{O}(\cdot)$ and $\tilde{\Theta}(\cdot)$ the variants of the big-O notation that hide polylogarithmic factors.

A Markov Decision Process (MDP) is a tuple $M = (\mathcal{S}, \mathcal{A}, Q)$, where $\mathcal{S}$ is a measurable state space, $\mathcal{A}$ is a finite action space, and $Q : \mathcal{S} \times \mathcal{A} \to \mathcal{M}_1([0,1] \times \mathcal{S})$ is the transition-reward kernel. We

define the transition and reward distributions $P : \mathcal{S} \times \mathcal{A} \to \mathcal{M}_1(\mathcal{S})$ and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathcal{M}_1([0,1])$ as the marginals of Q. By a slight abuse of notation, for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$, let us denote by $P(\cdot|s,a)$ and $\mathcal{R}(\cdot|s,a)$ denote the distributions $P(s,a)$ and $\mathcal{R}(s,a)$, respectively. Let us further denote by $r(s,a) = \int_0^1 x \, \mathrm{d}\mathcal{R}(x|s,a)$ the expected reward for a state and action. In the discounted episodic setting, we assume that there is a designated start state $s_0 \in \mathcal{S}$.

Starting from any state $s \in \mathcal{S}$, a stationary memoryless policy $\pi : \mathcal{S} \to \mathcal{M}_1(\mathcal{A})$ interacts with the MDP in a sequential manner for time-steps $t \in \mathbb{N}$, defining a probability distribution $\mathcal{P}_{\pi,s}$ over the episode trajectory $\{S_i, A_i, R_i\}_{i \in \mathbb{N}}$ as follows: $S_0 = s$ deterministically, $A_i \sim \pi(S_i)$, and $(R_i, S_{i+1}) \sim Q(S_i, A_i)$. By a slight variation, let us denote by $\mathcal{P}_{\pi,s,a}$ (for some $a \in \mathcal{A}$) the distribution of the trajectory when $A_0 = a$ deterministically, while the distribution of the rest of the trajectory is defined analogously.

This allows us to conveniently define the expected state-value and action-value functions for some discount factor $0 < \gamma < 1$ as

$$v^\pi(s) = \mathbb{E}_{\pi,s} \sum_{t \in \mathbb{N}} \gamma^t R_t \quad \text{and} \quad q^\pi(s,a) = \mathbb{E}_{\pi,s,a} \sum_{t \in \mathbb{N}} \gamma^t R_t \qquad \text{for all } s, a \in \mathcal{S} \times \mathcal{A}\,, \tag{2}$$

where throughout the paper we use the convention that $\mathbb{E}_\bullet$ is the expectation operator corresponding to a distribution $\mathcal{P}_\bullet$ (e.g., $\mathbb{E}_{\pi,s}$ is the expectation with respect to $\mathcal{P}_{\pi,s}$). It is well-known that there exists an optimal deterministic policy $\pi^\star$ such that:

$$\sup_\pi v^\pi(s) = v^{\pi^\star}(s) \qquad \text{and} \qquad \sup_\pi q^\pi(s,a) = q^{\pi^\star}(s,a) \qquad \text{for all } s, a \in \mathcal{S} \times \mathcal{A}\,.$$

Let $v^\star = v^{\pi^\star}$ and $q^\star = q^{\pi^\star}$. For any policy π, v^π and q^π are known to satisfy the Bellman equations [Puterman, 1994]:

$$v^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q^\pi(s,a) \;\text{ and }\; q^\pi(s,a) = r(s,a) + \gamma \int_{s' \in \mathcal{S}} v^\pi(s') \, \mathrm{d}P(s'|s,a) \quad \text{for all } s, a \in \mathcal{S} \times \mathcal{A}\,. \tag{3}$$

Finally, we call a policy π deterministic if for all states, $\pi(s)$ is a distribution that assigns unit weight to one action and zero weight to the others. For notational simplicity, for a deterministic policy π, we denote by $\pi(s)$ the action π chooses (deterministically) in state $s \in \mathcal{S}$. We assume that $\arg\max_{a \in \mathcal{A}}$ breaks ties in a consistent manner by ordering the actions (using the notation $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_{|\mathcal{A}|})$) and always choosing action $\mathcal{A}_i$ with the lowest index i that achieves the maximum.

3 Confident Approximate Policy Iteration

The purpose of this section is to introduce Confident Approximate Policy Iteration (CAPI), our new approximate dynamic programming algorithm. In approximate dynamic programming, the methods are designed around oracles that return either an approximation to the application of the Bellman optimality operator to a value function ("approximate value iteration"), or an oracle that returns an approximation to the value function of some policy ("approximate policy iteration"). Our setting is the second. The novelty is that we assume access to the accuracy of the approximation and use this knowledge to modify the policy update which will lead to the improved guarantees.

We present the pseudocodes of API [Bertsekas and Tsitsiklis, 1996] and CAPI jointly in Algorithm 1; starting from an arbitrary (deterministic) policy π_0, the algorithm iterates a policy estimation (Line 2) and a policy update step (Line 3) I times. The policy update for API is greedy with respect to the action-value estimates $\hat{q}$ and is defined as $\pi_{\hat{q}}(s) = \arg\max_{a \in \mathcal{A}} \hat{q}(s,a)$. For CAPI, the policy update further relies on a global estimation-accuracy parameter ω (we omit this dependence), and a set of fixed-states $\mathcal{S}_{\text{fix}} \subseteq \mathcal{S}$. For the purposes of this section, it is enough to keep $\mathcal{S}_{\text{fix}} = \{\}$. CAPI updates the policy to one that acts greedily with respect to $\hat{q}$ *only* on states not in $\mathcal{S}_{\text{fix}}$ where it is confident that this leads to an improvement over the previous policy (Case 4a); otherwise, the new policy will return the same action as the previous one (Case 4b). To decide, $\hat{q}(s, \pi(s)) + \omega$ is treated as the upper bound on the previous policy's value, and $\max_{a \in \mathcal{A}} \hat{q}(s,a) - \omega$ as the lower bound of the action-value of the greedy action:

$$\pi_{\hat{q},\pi,\mathcal{S}_{\text{fix}}}(s) = \begin{cases} \pi(s)\,, & \text{if } s \in \mathcal{S}_{\text{fix}} \text{ or } \hat{q}(s,\pi(s)) + \omega \geq \max_{a \in \mathcal{A}} \hat{q}(s,a) - \omega\;; \quad \text{(4a)} \\ \arg\max_{a \in \mathcal{A}} \hat{q}(s,a)\,, & \text{otherwise}\,. \quad \text{(4b)} \end{cases}$$

Algorithm 1 API and CAPI

1: **for** $i = 1$ to I **do**
2: $\quad \hat{q} \leftarrow \text{Estimate}(\pi_{i-1})$
3: $\quad \pi_i \leftarrow \begin{cases} \pi_{\hat{q}} & \text{API} \\ \pi_{\hat{q},\pi_{i-1},\mathcal{S}_{\text{fix}}} & \text{CAPI} \end{cases}$
4: **return** π_I

Let π be any deterministic memoryless policy, $\hat{q}^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be any function, $\omega \in \mathbb{R}_+$, and $\mathcal{S}_{\text{fix}} \subseteq \mathcal{S}$. First, we show that as long as $\hat{q}$ is ω-accurate, the CAPI policy update only improves the policy's values:

Lemma 3.1 (No deterioration)**.** *Let $\pi' = \pi_{\hat{q}^\pi,\pi,\mathcal{S}_{fix}}$. Assume that for all $s \in \mathcal{S} \setminus \mathcal{S}_{fix}$ and $a \in \mathcal{A}$, $\hat{q}^\pi(s,a) \approx_\omega q^\pi(s,a)$. Then, for any $s \in \mathcal{S}$,*

$$v^{\pi'}(s) \geq v^\pi(s)\,.$$

Proof. Fix any $s \in \mathcal{S}$. If $s \in \mathcal{S}_{\text{fix}}$ or $\hat{q}(s,\pi(s)) + \omega \geq \max_{a\in\mathcal{A}} \hat{q}(s,a) - \omega$, then $\pi'(s) = \pi(s)$ and therefore $q^\pi(s,\pi'(s)) = v^\pi(s)$. Otherwise, $s \notin \mathcal{S}_{\text{fix}}$ and $\hat{q}(s,\pi(s)) + \omega \leq \max_{a\in\mathcal{A}} \hat{q}(s,a) - \omega$, hence $\pi'(s) = \arg\max_{a\in\mathcal{A}} \hat{q}^\pi(s,a)$ and by our assumptions $q^\pi(s,\pi'(s)) \geq \max_{a\in\mathcal{A}} \hat{q}(s,a) - \omega > \hat{q}(s,\pi(s)) + \omega \geq v^\pi(s)$. Therefore, in any case, $q^\pi(s,\pi'(s)) \geq v^\pi(s)$. Since this holds for any $s \in \mathcal{S}$, the Policy Improvement Theorem [Sutton and Barto, 2018, Section 4.2] implies that for any $s \in \mathcal{S}$, $v^{\pi'}(s) \geq v^\pi(s)$. □

Next we introduce two approximate optimality criterion for a policy on a set of states:

Definition 3.2 (Policy optimality on a set of states)**.** *We say that a policy π is Δ-optimal (for some $\Delta \geq 0$) on a set of states $\mathcal{S}' \subseteq \mathcal{S}$, if for all $s \in \mathcal{S}'$,*

$$v^\star(s) - v^\pi(s) \leq \Delta\,. \tag{5}$$

Definition 3.3 (Next-state optimality on a set of states)**.** *We say that a policy π is next-state Δ-optimal (for some $\Delta \geq 0$) on a set of states $\mathcal{S}' \subseteq \mathcal{S}$, if for all $s \in \mathcal{S}'$ and all actions $a \in \mathcal{A}$,*

$$\int_{s'\in\mathcal{S}} \left(v^\star(s') - v^\pi(s')\right) \mathrm{d}P(s'|s,a) \leq \Delta\,.$$

Note that in the special case of $\mathcal{S}' = \mathcal{S}$, if π is Δ-optimal on $\mathcal{S}$, then it is also next-state Δ-optimal on $\mathcal{S}$. Next, we show that the suboptimality of a policy updated by CAPI evolves as follows (the proof is relegated to Appendix A):

Lemma 3.4 (Iteration progress)**.** *Let $\pi' = \pi_{\hat{q}^\pi,\pi,\mathcal{S}_{fix}}$. Assume that for all $s \in \mathcal{S} \setminus \mathcal{S}_{fix}$ and $a \in \mathcal{A}$, $\hat{q}^\pi(s,a) \approx_\omega q^\pi(s,a)$, and that π is next-state Δ-optimal on $\mathcal{S} \setminus \mathcal{S}_{fix}$. Then π' is $(4\omega + \gamma\Delta)$-optimal on $\mathcal{S} \setminus \mathcal{S}_{fix}$.*

3.1 CAPI guarantee with accurate estimation everywhere

To obtain a final suboptimality guarantee for CAPI, first consider the ideal scenario in which we assume that we have a mechanism to estimate $q^\pi(s,a)$ up to some ω accuracy for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, and for any policy π:

Assumption 3.5. *There is an oracle called Estimate that accepts a policy π and returns $\hat{q}^\pi : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ such that for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, $\hat{q}^\pi(s,a) \approx_\omega q^\pi(s,a)$.*

Theorem 3.6 (CAPI performance)**.** *Let $\mathcal{S}_{fix} = \{\}$, set the iteration count to $I = \lceil \log\omega/\log\gamma \rceil$, and assume that the estimation used in Line 2 satisfies Assumption 3.5. Then CAPI (Algorithm 1) returns with a policy π_I that is $5\omega/(1-\gamma)$-optimal on $\mathcal{S}$.*

Proof. We prove by induction that policy π_i is Δ_i-optimal on $\mathcal{S}$ for

$$\Delta_i = 4\omega\left(\sum_{j\in[i]} \gamma^j\right) + \frac{\gamma^i}{1-\gamma}\,.$$

This holds immediately for the base case of $i = 0$, as rewards are bounded in $[0, 1]$ and thus $v^\star(s) \leq 1/(1-\gamma)$ for any s. Assuming now that the inductive hypothesis holds for $i-1$ we observe that π_{i-1} is next-state Δ-optimal on $\mathcal{S} = \mathcal{S} \setminus \mathcal{S}_{\text{fix}}$. Together with Assumption 3.5, we can use Lemma 3.4 to derive that $v^\star(s) - v^{\pi_i}(s) \leq 4\omega + \gamma\Delta_{i-1} = \Delta_i$, which finishes the induction. Finally, we observe that for $i = I$, this implies that π_I is Δ_I-optimal with $\Delta_I \leq \frac{4\omega}{1-\gamma} + \frac{\gamma^I}{1-\gamma} \leq \frac{5\omega}{1-\gamma}$. □

4 Local access planning with q^π-realizability

Our planner, CAPI-QPI-PLAN, is based on CONFIDENT MC-LSPI of Yin et al. [2022]. They gradually build a "core-set" of state-action pairs whose corresponding features are informative. They measure the q-values on them using rollouts. The procedure is restarted with an extended core-set whenever the algorithm encounters a new informative feature. If such a new feature is not encountered, the estimation error can be controlled. The estimation is then extended to all state-action pairs using the least-squares estimator. Finally, the extended estimation is used for API.

Our planner improves on CONFIDENT MC-LSPI in two ways. First, using CAPI instead of API, improves the final suboptimality bound by a factor of the effective horizon. Second, we apply a novel analysis on a more modular variant of their CONFIDENTROLLOUT subroutine, that delivers q-estimation accuracy guarantees with respect to a large class of policies simultaneously. This allows for a dynamically evolving version of policy iteration, that does not have to restart whenever a new informative feature is encountered. Intuitively, this prevents duplication of work.

4.1 Estimation oracle

Towards finding an algorithm that works on the basis of planning with local access and whose performance degrades gracefully with the uniform approximation error, we must weaken Assumption 3.5. This is because under local access, we cannot guarantee to cover all states, or hope to obtain an accurate estimation for all states. Instead, we are interested in an accuracy guarantee that holds for q-values only on some subset $\mathcal{S}' \subseteq \mathcal{S}$ of states, but holds simultaneously for *any* policy that agrees with π on $\mathcal{S}'$ but may take arbitrary values elsewhere. For this, we define the extended set of policies:

Definition 4.1. *Let Π_{det} be the set of all deterministic stationary memoryless policies, $\pi \in \Pi_{det}$, and $\mathcal{S}' \subseteq \mathcal{S}$. For $(\pi, \mathcal{S}')$, we define $\Pi_{\pi,\mathcal{S}'}$ to be the set of policies that agree with π on $s \in \mathcal{S}'$:*

$$\Pi_{\pi,\mathcal{S}'} = \{\pi' \in \Pi_{det} \,:\, \forall s \in \mathcal{S}',\, \pi(s) = \pi'(s)\}$$

We aim to first accurately estimate $q^\pi(s, a)$ for *some specific* (s, a) pairs, based on which we will extend the estimation to other state-action pairs using least-squares. To this end, we first devise a subroutine called Measure (Algorithm 2). Measure is a modularized variant of the CONFIDENTROLLOUT subroutine of Yin et al. [2022]. The modularity of our variant is due to the parameter $\mathcal{S}'$ that corresponds to the set of states on which the planner is "confident" for CONFIDENTROLLOUT. Measure unrolls the policy π starting from (s, a) for a number of episodes each lasting H steps, and returns with the average measured reward. Throughout, we let $H = \lceil \log(\omega/4(1-\gamma))/\log\gamma \rceil$ be the effective horizon. At the end of this process, Measure returns status *success* along with the empirical average q-value, where compared to Eq. (2), the discounted summation of rewards is truncated to the length of the effective horizon. If however, during this process, the algorithm ever encounters a state not in its input $\mathcal{S}'$, it returns with status *discover*, along with that state. This is because in such cases, the algorithm could no longer guarantee an accurate estimation with respect to any member of the extended set of policies.

Lemma 4.2. *For any input parameters $s, a, \pi, \mathcal{S}', \omega, \zeta$, Measure$(s, a, \pi, \mathcal{S}', \omega, \zeta)$ either returns with $(\textit{discover}, s')$ for some $s' \notin \mathcal{S}'$ (Line 6), or it returns with $(\textit{success}, \tilde{q})$ such that with probability at least $1 - \zeta$,*

$$q^{\pi'}(s, a) \approx_\omega \tilde{q} \quad \textit{for all} \quad \pi' \in \Pi_{\pi,\mathcal{S}'}. \tag{6}$$

The proof is given in Appendix B.

Suppose we have a list of state-action pairs $C = (s_i, a_i)_{i \in [|C|]}$ and corresponding q-estimates $\bar{q} = (\bar{q}_i)_{i \in |C|}$. We use the regularized least-squares estimator LSE (Eq. 8) to extend the estimates for

Algorithm 2 Measure

Input: state s, action a, deterministic policy π, set of states $\mathcal{S}' \subseteq \mathcal{S}$, accuracy $\omega > 0$, failure probability $\zeta \in (0, 1]$
Initialize: $H \leftarrow \lceil \log(\omega/4(1-\gamma))/\log\gamma \rceil$, $n \leftarrow \left\lceil (\omega/4)^{-2}(1-\gamma)^{-2}\log(2/\zeta)/2 \right\rceil$
for $i = 1$ to n **do**
  $(S, R_{i,0}) \leftarrow \text{Simulator}(s, a)$
  for $h = 1$ to $H - 1$ **do**
    if $S \notin \mathcal{S}'$ **then return** (discover, S)
    $A \leftarrow \pi(S)$
    $(S, R_{i,h}) \leftarrow \text{Simulator}(S, A)$ ▹ Call to the simulator oracle
return (success, $\frac{1}{n}\sum_{i=1}^{n}\sum_{h=0}^{H-1}\gamma^h R_{i,h}$)

all state-action pairs, with regularization parameter $\lambda = \omega^2/B^2$:

$$V(C) = \lambda\mathbb{I} + \sum_{i\in[|C|]} \varphi(s_i, a_i)\varphi(s_i, a_i)^\top , \tag{7}$$

$$\text{LSE}_{C,\bar{q}}(s, a) = \left\langle \varphi(s, a), V(C)^{-1} \sum_{i\in[|C|]} \varphi(s_i, a_i)\bar{q}_i \right\rangle . \tag{8}$$

For $C = \bar{q} = ()$ (the empty sequence), we define $\text{LSE}_{C,\bar{q}}(\cdot,\cdot) = 0$. This estimator satisfies the guarantee below.

Lemma 4.3. *Let π be a memoryless stationary policy. Let $C = (s_i, a_i)_{i\in[n]}$ be sequences of state-action pairs of some length $n \in \mathbb{N}$ and $\bar{q} = (\bar{q}_i)_{i\in[n]}$ a sequence of corresponding reals such that for all $i \in [n]$, $q^\pi(s_i, a_i) \approx_\omega \bar{q}_i$. Then, for all $s, a \in \mathcal{S} \times \mathcal{A}$,*

$$\left|LSE_{C,\bar{q}}(s, a) - q^\pi(s, a)\right| \le \varepsilon + \|\varphi(s, a)\|_{V(C)^{-1}} \left(\sqrt{\lambda}B + (\omega + \varepsilon)\sqrt{n}\right) , \tag{9}$$

where ε is the uniform approximation error from Definition 1.1.

The proof is given in Appendix C. The order of the estimation accuracy bound (Eq. (9)) is optimal, as shown by the lower bounds of Du et al. [2019], Lattimore et al. [2020].

We intend to only use the LSE estimator in Eq. (8) and the bound in Lemma 4.3 for state-action pairs where $\|\varphi(s, a)\|_{V(C)^{-1}} \le 1$ (and $n = \tilde{O}(d)$). We call these state-action pairs *covered* by C, and we call states covered by C if all their corresponding actions are covered (Eqs. 10 and 11):

$$\text{ActionCover}(C) = \{s, a \in \mathcal{S} \times \mathcal{A} \,:\, \|\varphi(s, a)\|_{V(C)^{-1}} \le 1\} \tag{10}$$

$$\text{Cover}(C) = \{s \in \mathcal{S} \,:\, \forall a \in \mathcal{A},\ (s, a) \in \text{ActionCover}(C)\} . \tag{11}$$

We will use the parameter $\mathcal{S}_{\text{fix}}$ of CAPI (see CAPI's update rule in Eq. 4) to ensure policies are only updated on covered states, where the approximation error is small.

4.2 Main algorithm

Finally, we are ready to introduce CAPI-QPI-PLAN, the algorithm that handles the q^π-realizable case with local access planning (Algorithm 3). For this, we define levels $l = 0, 1, \ldots, H$, and corresponding suboptimality requirements: for any $l \in [H + 1]$,

$$\Delta_l = 8(\varepsilon + \omega)\left(\sqrt{\tilde{d}} + 1\right)\left(\sum_{j\in[l]} \gamma^j\right) + \frac{\gamma^l}{1-\gamma} ,$$

for $\tilde{d} = \tilde{\Theta}(d)$ defined in Eq. (13). For each level l, the algorithm maintains a policy π_l, and a set of "covered" states on which it can guarantee that π_l is a Δ_l-optimal policy. More specifically, this set is $\text{Cover}(C_l)$, where C_l is a list if state-action pairs with elements $C_{l,i} = (s_l^i, a_l^i)$ for $i \in [|C_l|]$. The algorithm maintains the following suboptimality guarantee below, which we prove in Appendix E after showing some further key properties of the algorithm.

Lemma 4.4. *Assuming that Eq. (6) holds whenever Measure returns* success*, π_l is Δ_l-optimal on Cover(C_l) (Definition 3.2) for all $l \in [H+1]$ throughout the execution of Algorithm 3.*

The algorithm aims to improve the policies, while "propagating" the members of C_l to C_{l+1}, and so on, all the way to C_H. During this, whenever the algorithm discovers a state-action pair with a sufficiently "new" feature direction, this pair is appended to the sequence C_0 corresponding to level 0, as there are no suboptimality guarantees yet available for such a state. However, such a discovery can only happen $\tilde{O}(d)$ times. When, eventually, all discovered state-action pairs end up in C_H, the final suboptimality guarantee is reached, and the algorithm returns with the final policy. Note that in the local access setting, the algorithm cannot enumerate the set Cover(C_l), but can decide membership ($s \in$ Cover(C_l)) for any $s \in \mathcal{S}$ it encounters. The algorithm maintains sequences $\bar{q}_l$, corresponding to C_l, for each level l. Whenever a new (s, a) pair is appended to the sequence C_l, a corresponding $\perp$ symbol is appended the sequence $\bar{q}_l$, to signal that an estimate of $q^{\pi_l}(s, a)$ is not yet known.

Algorithm 3 CAPI-QPI-PLAN

1: **Input:** start state $s_0 \in \mathcal{S}$, dimensionality d, parameter bound B, accuracy ω, failure probability $\delta > 0$
2: **Initialize:** $H \leftarrow \lceil \log(\omega/4(1-\gamma))/\log\gamma \rceil$, for $l \in [H+1]$, $C_l \leftarrow ()$, $\bar{q}_l \leftarrow ()$, $\pi_l \leftarrow$ policy that always returns action $\mathcal{A}_0$, $\lambda \leftarrow \omega^2/B^2$
3: **while** True **do**
4: **if** $\exists a \in \mathcal{A},\ (s_0, a) \notin$ ActionCover(C_0) **then**
5: append (s_0, a) to C_0, append $\perp$ to $\bar{q}_0$
6: **break**
7: let ℓ be the smallest integer such that $\bar{q}_{\ell,m} = \perp$ for some m ; set $\ell = H$ if no such l exists
8: **if** $\ell = H$ **then return** π_H
9: (status, result) $\leftarrow$ Measure$(s^m_\ell, a^m_\ell, \pi_\ell, \text{Cover}(C_\ell), \omega, \delta/(\tilde{d}H))$ ▹ recall $C_{\ell,m} = (s^m_\ell, a^m_\ell)$
10: **if** status = discover **then**
11: append (result, a) to C_0 for some a such that (result, a) $\notin$ ActionCover(C_0)
12: append $\perp$ to $\bar{q}_0$
13: **break**
14: $\bar{q}_{\ell,m} \leftarrow$ result
15: **if** $\nexists m'$ such that $\bar{q}_{\ell,m'} = \perp$ **then**
16: $\hat{q} \leftarrow \text{LSE}_{C_\ell, \bar{q}_\ell}$
17: $\pi' \leftarrow \pi_{\hat{q}, \pi_\ell, \mathcal{S} \setminus \text{Cover}(C_\ell)}$
18: $\pi_{\ell+1} \leftarrow (s \mapsto \pi_{\ell+1}(s)$ if $s \in$ Cover$(C_{\ell+1})$ else $\pi'(s))$
19: **for** $(s, a) \in C_\ell$ such that $(s, a) \notin C_{\ell+1}$ **do**
20: append (s, a) to $C_{\ell+1}$, $\perp$ to $\bar{q}_{\ell+1}$

After initializing C_0 to cover the starting state s_0 (Lines 4 to 6), the algorithm measures $q^{\pi_\ell}(s, a)$ for the smallest level ℓ for which there still exists a $\perp$ in the corresponding $\bar{q}_\ell$. After a successful measurement, if there are no more $\perp$s left at this level (i.e., in $\bar{q}_\ell$), the algorithm executes a policy update on π_ℓ (Line 17) using the least-squares estimate obtained from the measurements at this level, but only for states in Cover(C_ℓ) (using $\mathcal{S}_{\text{fix}} = \mathcal{S} \setminus$ Cover(C_ℓ)). Following this, Line 18 merges this new policy π' with the existing policy $\pi_{\ell+1}$ of the next level, setting $\pi_{\ell+1}$ to be the policy π'' defined as

$$\pi''(s) = \begin{cases} \pi_{\ell+1}(s) & \text{if } s \in \text{Cover}(C_{\ell+1}); \\ \pi'(s) & \text{otherwise.} \end{cases}$$

This ensures that the existing policy $\pi_{\ell+1}$ remains unchanged by π'' (its replacement) on states that are already covered by $C_{\ell+1}$, and therefore $\pi'' \in \Pi_{\pi_{\ell+1}, \text{Cover}(C_{\ell+1})} = \Pi_{\pi'', \text{Cover}(C_{\ell+1})}$. We also observe that C_l can only grow for any l (elements are never removed from these sequences), thus for any update where C_l is assigned a new value C'_l (Lines 5, 11, and 20), Cover$(C'_l) \supseteq$ Cover(C_l) and $\Pi_{\pi_l, \text{Cover}(C'_l)} \subseteq \Pi_{\pi_l, \text{Cover}(C_l)}$. Combining these properties yields the following result:

Lemma 4.5. *If for any $l \in [H]$, π_l and C_l take some values π^{old}_l and C^{old}_l at any point in the execution of the algorithm, then at any later point during the execution, $\pi_l \in \Pi_{\pi^{old}_l, Cover(C^{old}_l)}$.*

Any value in $\bar{q}_l$ that is set to anything other than $\perp$ will never change again. Combined with Lemma 4.5, we get that the accuracy guarantees of Lemma 4.2 continue to hold throughout:

Lemma 4.6. *Assuming that Eq. (6) holds whenever Measure returns* success*, for any level l and index m such that $\bar{q}_{l,m} \neq \perp$, $q^{\pi'}(s_l^m, a_l^m) \approx_\omega \bar{q}_{l,m}$ for all $\pi' \in \Pi_{\pi_l, Cover(C_l)}$ throughout the execution of Algorithm 3.*

Once $\pi_{\ell+1}$ is updated in Line 18, in Line 20 we append to the sequence $C_{\ell+1}$ all members of C_ℓ that are not yet in $C_{\ell+1}$, while adding a corresponding $\perp$ to $\bar{q}_{\ell+1}$ indicating that these q-values are not yet measured for policy $\pi_{\ell+1}$. Thus, whenever all $\perp$ values disappear from some level $l \in [H+1]$, by the end of that iteration $\text{ActionCover}(C_l) = \text{ActionCover}(C_{l+1})$. Together with the fact that for any $l \in [H+1]$, whenever C_l is extended, $\bar{q}_l$ is extended with a $\perp$, we have by induction that

Lemma 4.7. *Throughout the execution of Algorithm 3, after Line 7 when ℓ is set,*

$$ActionCover(C_0) = ActionCover(C_1) = \cdots = ActionCover(C_l)\,.$$

As a result, whenever the Measure call of Line 9 outputs (discover, s) for some state s, by Lemma 4.2, there is an action $a \in \mathcal{A}$ such that $(s,a) \notin \text{ActionCover}(C_l) = \text{ActionCover}(C_0)$. This explains why adding such an (s,a) pair to C_0 is always possible in Line 11. Consider the ith time Line 11 is executed, and denote s by s_i and a by a_i, and $V_i = \lambda\mathbb{I} + \sum_{t=1}^{i-1} \varphi(s_t,a_t)\varphi(s_t,a_t)^\top$. Observe that as $V_i = V(C)$, $(s_i,a_i) \notin \text{ActionCover}(C_0)$ implies $\|\varphi(s_i,a_i)\|_{V_i^{-1}} > 1$. Therefore, $\sum_{t=1}^{i} \min\{1, \|\varphi(s_t,a_t)\|_{V_t^{-1}}\} = i$, and thus by the elliptical potential lemma [Lattimore and Szepesvári, 2020, Lemma 19.4], $i \leq 2d \log\left(\frac{d\lambda + iL^2}{d\lambda}\right)$. This inequality is satisfied when substituting in i the total number of times Measure returns with *discover*. Since any element of C_l is also an element of C_0 for any $l \in [H+1]$, we have that at any time during the execution of Algorithm 3,

$$|C_l| \leq 4d \log\left(e + \frac{4L^2}{\lambda}\right) =: \tilde{d} = \tilde{O}(d)\,. \tag{13}$$

When Algorithm 3 returns at Line 8 with the policy π_H, it is Δ_H-optimal on $\text{Cover}(C_H)$ by Lemma 4.4 when the estimates of Measure are correct. Furthermore, $s_0 \in \text{Cover}(C_0)$ is guaranteed by Lines 4 to 6, and hence $s_0 \in \text{Cover}(C_H)$ by Lemma 4.7 when the algorithm finishes. Hence, bounding Δ_H using the definition of H immediately gives the following result:

Lemma 4.8. *Assuming that Eq. (6) holds whenever Measure returns* success*, the policy π returned by Algorithm 3 is Δ-optimal on $\{s_0\}$ for*

$$\Delta = 9(\varepsilon+\omega)\left(\sqrt{\tilde{d}}+1\right)(1-\gamma)^{-1} = \tilde{O}\left((\varepsilon+\omega)\sqrt{d}(1-\gamma)^{-1}\right)\,.$$

We conclude the paper by proving Theorem 1.2.

Proof of Theorem 1.2. Both the total failure probability and query complexity of Algorithm 3 depend on the number of times Measure is executed, as this is the only source of randomness and of interaction with the simulator. Measure can return *discover* at most $|C_0|$ times, which is bounded by $\tilde{d}$ by Eq. (13). For every $l \in [H]$, Measure is executed exactly once with returning *success* for each element of C_l. Hence, by Eq. (13) again, Measure returns *success* at most $\tilde{d}H$ times, each satisfying Eq. (6) with probability at least $1-\zeta = 1-\delta/(\tilde{d}H)$ by Lemma 4.2. By the union bound, Measure returns *success* in all occasions with probability at least $1-\delta$. Hence Eq. (6) holds with probability at least $1-\delta$, which, combined with Lemma 4.8, proves Eq. (1).

Each successful run of Measure executes at most nH queries (n is set in Line 2 of Algorithm 2). By noting that $H < (1-\gamma)^{-1}\log(4\omega^{-1}(1-\gamma)^{-1}) = \tilde{O}((1-\gamma)^{-1})$, in total Algorithm 3 executes at most $\tilde{O}\left(d(1-\gamma)^{-4}\omega^{-2}\right)$ queries. As this happens at most $\tilde{d}H$, we obtain the desired bound on the query complexity. □

References

Y. Abbasi-Yadkori, P. Bartlett, K. Bhatia, N. Lazic, Cs. Szepesvári, and G. Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *ICML*, pages 3692–3702, 2019.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate dynamic programming*, volume II. 4 edition, 2012.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2019.

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.

T. Lattimore and Cs. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Tor Lattimore, Csaba Szepesvári, and Gellért Weisz. Learning with good feature representations in bandits and in RL with a generative model. In *ICML*, pages 9464–9472, 2020.

A Woodbury Max. Inverting modified matrices. In *Memorandum Rept. 42, Statistical Research Group*, page 4. Princeton Univ., 1950.

Remi Munos. Error bounds for approximate policy iteration. In *ICML*, pages 560–567, 2003.

Remi Munos. Error bounds for approximate value iteration. In *AAAI*, pages 1006–1011, 2005.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

Daniel Russo. Approximation benefits of policy gradient methods with aggregated states. *arXiv preprint arXiv:2007.11684*, 2020.

Bruno Scherrer and Boris Lesner. On the use of non-stationary policies for stationary infinite-horizon Markov decision processes. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, 2012.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Andrew Wagenmaker, Yifang Chen, Max Simchowitz, Simon S Du, and Kevin Jamieson. Reward-free RL is no harder than reward-aware RL in linear Markov decision processes. *arXiv preprint arXiv:2201.11206*, 2022.

Yuanhao Wang, Ruosong Wang, and Sham Kakade. An exponential lower bound for linearly realizable MDP with constant suboptimality gap. *Advances in Neural Information Processing Systems*, 34, 2021.

Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in MDPs with linearly-realizable optimal action-value functions. In *ALT*, volume 132 of *Proceedings of Machine Learning Research*, pages 1237–1264, 2021.

Dong Yin, Botao Hao, Yasin Abbasi-Yadkori, Nevena Lazić, and Csaba Szepesvári. Efficient local planning with linear function approximation. In *International Conference on Algorithmic Learning Theory*, pages 1165–1192. PMLR, 2022.

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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]