Reward-Free Exploration for Reinforcement Learning

Chi Jin [1] **Akshay Krishnamurthy** [2] **Max Simchowitz** [3] **Tiancheng Yu** [4]

Abstract

Exploration is widely regarded as one of the most challenging aspects of reinforcement learning (RL), with many naive approaches succumbing to exponential sample complexity. To isolate the challenges of exploration, we propose a new "reward-free RL" framework. In the exploration phase, the agent first collects trajectories from an MDP $\mathcal{M}$ *without* a pre-specified reward function. After exploration, it is tasked with computing near-optimal policies under for $\mathcal{M}$ for a collection of given reward functions. This framework is particularly suitable when there are many reward functions of interest, or when the reward function is shaped by an external agent to elicit desired behavior.

We give an efficient algorithm that conducts $\tilde{\mathcal{O}}(S^2A\mathrm{poly}(H)/\epsilon^2)$ episodes of exploration and returns ϵ-suboptimal policies for an *arbitrary* number of reward functions. We achieve this by finding exploratory policies that visit each "significant" state with probability proportional to its maximum visitation probability under *any* possible policy. Moreover, our planning procedure can be instantiated by any black-box approximate planner, such as value iteration or natural policy gradient. We also give a nearly-matching $\Omega(S^2AH^2/\epsilon^2)$ lower bound, demonstrating the near-optimality of our algorithm in this setting.

1. Introduction

In reinforcement learning (RL), an agent repeatedly interacts with an unknown environment with the goal of maximizing its cumulative reward. To do so, the agent must engage in *exploration*, learning to visit states in order to investigate whether they hold high reward.

[1]Princeton University [2]Microsoft Research, New York [3]University of California, Berkeley [4]Massachusetts Institute of Technology. Correspondence to: Chi Jin <chij@princeton.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Exploration is widely regarded as the most significant challenge in RL, because the agent may have to take precise sequences of actions to reach states with high reward. Here, simple randomized exploration strategies provably fail: for example, a random walk can take exponential time to reach the corner of the environment where the agent can accummulate high reward (Li, 2012). While reinforcement learning has seen a tremendous surge of recent research activity, essentially all of the standard algorithms deployed in practice employ simple randomization or its variants, and consequently incur extremely high sample complexity.

On the other hand, sophisticated exploration strategies which deliberately incentivize the agent to visit new states are provably sample-efficient (c.f., Kearns & Singh (2002); Brafman & Tennenholtz (2002); Azar et al. (2017); Dann et al. (2017); Jin et al. (2018)), with recent work providing a nearly-complete theoretical understanding for maximizing a single prespecified reward function (Dann & Brunskill, 2015; Azar et al., 2017; Zanette & Brunskill, 2019; Simchowitz & Jamieson, 2019). In practice, however, reward functions are often iteratively engineered to encourage desired behavior via trial and error (e.g. in constrained RL formulations (Altman, 1999; Achiam et al., 2017; Tessler et al., 2018; Miryoosefi et al., 2019)). In such cases, repeatedly invoking the same reinforcement learning algorithm with different reward functions can be quite sample inefficient.

One solution to avoid excessive data collection in such settings is to first collect a dataset with good coverage over all possible scenarios in the environment, and then apply a "Batch-RL" algorithm. Indeed many algorithms are known for computing near optimal policies from previously collected data, provided that the dataset has good coverage (Munos & Szepesvári, 2008; Antos et al., 2008; Chen & Jiang, 2019; Agarwal et al., 2019). However, prior work provides little guidance into how to obtain such good coverage.

In this paper, we aim to develop an end-to-end instantiation of this proposal. To this end we ask:

> How can we efficiently explore an environment without using any reward information?

In particular, by exploring the environment, we aim to

gather sufficient information so that we can compute the near-optimal policies for *any* reward function after-the-fact.

Our Contributions. In this paper, we present the first near-optimal upper and lower bounds which characterize the sample complexity of achieving provably sufficient coverage for Batch-RL. We do so by adopting a novel "reward-free RL" paradigm: During an exploration phase, the agent collects trajectories from an MDP $\mathcal{M}$ *without* a pre-specified reward function. Then, in a planning phase, it is tasked with computing near-optimal policies under the transitions of $\mathcal{M}$ for a large collection of given reward functions.

Letting S denote the number of states, A the number of actions, H the horizon, and ϵ the desired accuracy, we give an efficient algorithm which, after conducting $\tilde{\mathcal{O}}(S^2A\text{poly}(H)/\epsilon^2)$ episodes of exploration, collects a data set with sufficiently good coverage to enable application of standard Batch-RL solvers. Specifically, we show that when given a reward function r we can find an ϵ-suboptimal policy for the true MDP $\mathcal{M}$ with reward r, using the dataset alone and no additional data collection. This guarantee holds for all possible reward functions simultaneously, without needing to collect more data to ensure statistical correctness as new reward functions are considered.

Our exploration phase is conceptually simple, using an existing RL algorithm as a black-box (Zanette & Brunskill, 2019), and our planning phase accommodates arbitrary Batch-RL solvers. We instantiate our result with value iteration and natural policy gradient as special cases. By decoupling exploration and planning, our work sheds light on the algorithmic mechanisms required for sample efficient reinforcement learning. We hope that this insight will be useful in the design of provably efficient algorithms for more practically relevant RL settings, such as those where function approximation is required.

In addition to our algorithmic results, we establish a nearly-matching $\Omega(S^2AH^2/\epsilon^2)$ lower bound, demonstrating the near-optimality of our algorithm in this paradigm. Notably, this lower bound quantifies a price of "good-coverage" in the reward-free setting: while RL with a pre-specified reward has sample complexity of only $\widetilde{\Theta}(SAH^2/\epsilon^2)$ (Dann & Brunskill, 2015), the reward-free sample complexity is a factor of S larger.

Technical Novelty. The main technical challenge in our work involves handling environments with states that are difficult to reach. In such cases, we cannot learn the transition operator to high accuracy uniformly over the environment, simply because we cannot reach these states to collect enough data. With $\lambda(s)$ denoting the maximal probability of visiting state s under any policy, our key observation is that we can partition the state space into two groups: the states with $\lambda(s)$ so small that they have negligible contribution to reward optimization, and the rest. We introduce a rigorous analysis which enables us to "ignore" the difficult-to-visit states altogether and only requires that we visit the remaining states with probability proportional to $\lambda(s)$. To achieve this latter guarantee, we conduct our exploration with the EULER algorithm (Zanette & Brunskill, 2019), which in our context yields refined sample complexity guarantees in terms of $\lambda(s)$. We believe that this decomposition of states into their ease of being reached may be of broader interest. Our lower bound also adopts a novel and sophisticated construction, detailed in Section 4.

Related work. For reward-free exploration in the tabular setting, we are aware of only a few prior approaches. First, when one runs a PAC-RL algorithm like RMAX with no reward function (Brafman & Tennenholtz, 2002), it does visit the entire state space and can be shown to provide a coverage guarantee. However, for RMAX in particular the resulting sample complexity is quite poor, and significantly worse than our near-optimal guarantee (See Appendix A for a detailed calculation). We expect similar behavior from other PAC algorithms, because reward-dependent exploration is typically suboptimal for the reward-free setting.

Second, one can extract the exploration component of recent results for RL with function approximation (Du et al., 2019; Misra et al., 2019). Specifically, the former employs a model based approach where a model is iteratively refined by planning to visit unexplored states, while the latter uses model free dynamic programming to identify and reach all states. While these papers address a more difficult setting, it is relatively straightforward to specialize their results to the tabular setting. In this case, both methods guarantee coverage, but they have suboptimal sample complexity and require that all states can be visited with significant probability. In contrast, our approach requires no visitation probability assumptions and achieves the optimal sample complexity.

The last point of comparison is a recent result of Hazan et al. (2019), that gives an efficient algorithm for finding a certain exploratory policy. They use a Frank-Wolfe style algorithm to find a policy whose state occupancy measure has maximum entropy. One can show that an exact optimizer for their objective has a similar coverage property to our exploratory policy, but the Frank-Wolfe style algorithm can only guarantee an approximate optimizer. They do not analyze how the optimization error enters in the coverage guarantee, but we are able to show that setting the error to $O(1/S)$ suffices (see Appendix B). Unfortunately, this implies that their sample complexity scales with S^5, which is much worse than ours. (Lim & Auer, 2012; Tarbouriech & Lazaric, 2019) focus on a similar formulation using dif-

ferent optimization criterion. More generally, their result is not end-to-end in that they do not show how to use their policy for planning, and they do not establish a final sample complexity bound, both of which we do here.

Finally, the main source of motivation for our work is recent and classical results on batch reinforcement learning (Munos & Szepesvári, 2008; Antos et al., 2008; Chen & Jiang, 2019; Agarwal et al., 2019), a setting where the goal is to find a near optimal policy, given an *a priori* dataset collected by some logging policy that satisfies certain coverage properties. In this paper, we show how to find such a logging policy for the tabular setting, which enables straightforward application of these batch RL results. As an example, we show how to apply both value iteration and natural policy gradient to optimize the policy given any reward function. More generally, these works typically also consider the function approximation setting, and we believe our modular approach will facilitate development of provably efficient algorithms for these challenging settings.

2. Preliminaries

We consider the setting of a tabular episodic Markov decision process, $\mathrm{MDP}(\mathcal{S}, \mathcal{A}, \mathrm{H}, \mathbb{P}, \mathrm{r})$, where $\mathcal{S}$ is the set of states with $|\mathcal{S}| = S$, $\mathcal{A}$ is the set of actions with $|\mathcal{A}| = A$, H is the number of steps in each episode, $\mathbb{P}$ is the time-dependent transition matrix so that $\mathbb{P}_h(\cdot|s,a)$ gives the distribution over the next state if action a is taken from state s at step $h \in [H]$, and $r_h \colon \mathcal{S} \times \mathcal{A} \to [0,1]$ is the deterministic reward function at step h.[1] Note that we are assuming that rewards are in $[0,1]$ for normalization.

In each episode of a standard MDP, an initial state s_1 is picked from an unknown initial distribution $\mathbb{P}_1(\cdot)$. Then, at each step $h \in [H]$, the agent observes state $s_h \in \mathcal{S}$, picks an action $a_h \in \mathcal{A}$, receives reward $r_h(s_h, a_h)$, and then transitions to the next state s_{h+1}, which is drawn from the distribution $\mathbb{P}_h(\cdot|s_h, a_h)$. The episode ends after the H^{th} reward is collected.

A (non-stationary, stochastic) policy π is a collection of H functions $\left\{\pi_h : \mathcal{S} \to \Delta_{\mathcal{A}}\right\}_{h\in[H]}$, where $\Delta_{\mathcal{A}}$ is the probability simplex over action set $\mathcal{A}$. As notation, we use $\pi(\cdot|s)$ to denote the action distribution for policy π in state s. We use $V_h^\pi \colon \mathcal{S} \to \mathbb{R}$ to denote the value function at step h under policy π, which gives the expected sum of remaining rewards received under policy π, starting from $s_h = s$, until the end of the episode. That is,

$$V_h^\pi(s) := \mathbb{E}_\pi \left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) | s_h = s \right].$$

[1] While we study deterministic reward functions for notational simplicity, our results generalize to randomized reward functions.

Protocol 1 Reward-Free Exploration

for $k = 1$ **to** K **do**
  learner decides a policy π_k
  environment samples the initial state $s_0 \sim \mathbb{P}_1$.
  for $h = 1$ **to** H **do**
    learner selects action $a_h \sim \pi_h(\cdot|s_h)$
    environment transitions to $s_{h+1} \sim \mathbb{P}_h(\cdot|s_h, a_h)$
    learner observes the next state s_{h+1}
  end for
end for

Accordingly, we also define $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to denote action-value function at step h, so that $Q_h^\pi(s,a)$ gives the expected sum of remaining rewards received under policy π, starting from $s_h = s, a_h = a$, until the end of the episode. Formally:

$$Q_h^\pi(s,a) := \mathbb{E}_\pi \left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) | s_h = s, a_h = a \right].$$

Since the state and action spaces, and the horizon, are all finite, there always exists (see, e.g., (Azar et al., 2017)) an optimal policy $\pi^\star$ which gives the optimal value $V_h^\star(s) = \sup_\pi V_h^\pi(s)$ for all $s \in \mathcal{S}$ and $h \in [H]$. As notation, define $[\mathbb{P}_h V_{h+1}](s,a) := \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)} V_{h+1}(s')$. Recall the Bellman equation

$$V_h^\pi(s) = Q_h^\pi(s, \pi_h(s)), Q_h^\pi(s,a) = (r_h + \mathbb{P}_h V_{h+1}^\pi)(s,a) \tag{1}$$

and the Bellman optimality equation:

$$V_h^\star(s) = \max_{a \in \mathcal{A}} Q_h^\star(s,a), Q_h^\star(s,a) := (r_h + \mathbb{P}_h V_{h+1}^\star)(s,a). \tag{2}$$

where we define $V_{H+1}^\pi(s) = V_{H+1}^\star(s) = 0$ for any $s \in \mathcal{S}$.

The RL objective is to find an ϵ-optimal policy π, satisfying

$$\mathbb{E}_{s_1 \sim \mathbb{P}_1}[V_1^\star(s_1) - V_1^\pi(s_1)] \le \epsilon$$

Reward-free Exploration. In the reward-free setting, we would like to design algorithms that efficiently explore the state space without the guidance of reward information. Formally, the agent interacts with the environment through Protocol 1—a reward-free version of the MDP, where the agent can transit as usual but does not collect any rewards. Over the course of K episodes following Protocol 1, the agent collects a dataset of visited states, actions, and transitions $\mathcal{D} = \{s_h^{(k)}, a_h^{(k)}\}_{(k,h)\in[K]\times[H]}$, which is the outcome of the *exploration phase*.

The effectiveness of the exploration strategy is evaluated in the next phase—the *planning phase*—in which the agent is

no longer allowed to interact with the MDP. In this phase, the agent is given a reward function $r(\cdot,\cdot)$ that can be potentially adversarially designed, and the **objective** here is to compute a near optimal policy for this reward function using the dataset $\mathcal{D}$. Performance is measured in terms of how many episodes K are required in the exploration phase so that the agent can reliably achieve the objective above. As notation, we use $V(\cdot; r)$ to emphasize that the value function depends on the reward r.

We remark that providing the reward function after the exploration phase (as opposed to before) makes the setting more challenging, and so our algorithm applies to the easier setting. We also note that our results address the setting where the reward is observed through interaction with the environment, as learning the reward is typically not the statistical barrier to efficient RL. Indeed, a provably effective reward-free exploration strategy must visit all "significant" state-action pairs (see Definition 3.2) sufficiently many times anyway, and this experience is sufficient to learn the reward function.

3. Main Results

We are now ready to state our main theorem. It asserts that our algorithm, which we will describe in the subsequent sections, is a reward-free exploration algorithm with sample complexity $\tilde{\mathcal{O}}(H^5S^2A/\epsilon^2)$, ignoring lower order terms. In other words, after this many episodes interacting with the MDP via Protocol 1, our algorithm can compute ϵ-optimal policies for arbitrarily many reward functions. The theorem demonstrates that the sample complexity of reward-free exploration is at most $\tilde{\mathcal{O}}(H^5S^2A/\epsilon^2)$, which we will show to be near-optimal with our lower bound in the next section.

Theorem 3.1. *Ther exists an absolute constant $c > 0$ and a reward-free exploration algorithm such that, for any $p \in (0,1)$, with probability at least $1-p$, the algorithm outputs ϵ-optimal policies for an arbitrary number of adaptively chosen reward functions. The number of episodes collected in the exploration phase is bounded by*

$$c \cdot \left[\frac{H^5S^2A\iota}{\epsilon^2} + \frac{S^4AH^7\iota^3}{\epsilon} \right], \tag{3}$$

where $\iota := \log(SAH/(p\epsilon))$.

Notice when ϵ is small, $\tilde{\mathcal{O}}(S^4AH^7/\epsilon)$ is a lower order term. This term comes from the burn-in term in EULER, which is the current state of the art for the standard episodic RL formulation. Any improvements there would transfer to our guarantees, but whether the burn-in term is required remains an intriguing open problem.

We emphasize that the correctness guarantee here is quite strong: the dataset $\mathcal{D}$ collected by the algorithm is such that any number of adaptively chosen reward functions can be optimized with no further data collection. In contrast, if we naïvely deployed a reward-sensitive RL algorithm, we would have to collect additional trajectories for each reward function, which could be quite sample inefficient. We emphasize that requiring near-optimal policies for many reward functions is quite common in applications, especially when we design reward functions by trial and error to elicit specific behaviors.

Algorithm overview. Our algorithm proceeds with following high level steps:

1. learn a set of policies Ψ which allow us to visit all "significant" states with reasonable probability.
2. collect a sufficient amount of data by executing policies in Ψ.
3. compute the empirical transition matrix $\hat{\mathbb{P}}$ using the collected data.
4. for each reward function r, find a near-optimal policy by invoking a planning algorithm with transitions $\hat{\mathbb{P}}$ and reward r.

The first two steps are performed in the exploration phase, while the latter two steps are performed in the planning phase. In Section 3.1 and Section 3.2, we will present our formal algorithms and the corresponding theoretical guarantees for two phases separately. One important feature of our algorithm is that we can use existing approximate MDP solvers or batch-RL algorithms in the last step. We demonstrate with two examples, namely Value Iteration (VI) and Natural Policy Gradient (NPG), in Section 3.3.

3.1. Exploration Phase

The goal of exploration is to visit all possible states so that the agent can gather sufficient information in order to find the optimal policy eventually. However, rather different from the bandit setting where agent can select an arbitrary arm to pull, it is possible that certain state in the MDP is very difficult to reach no matter what policy the agent is taking. Therefore, we first introduce the concept of the state being "significant". See Figure 1 for illustrations.

Definition 3.2. A state s in step h is **δ-significant** if there exists a policy π, so that the probability to reach s following policy π is greater than δ. In symbol:

$$\max_{\pi} P_h^{\pi}(s) \geq \delta$$

Intuitively, with limited budget of samples and runtime, one can only hope to visit all significant states. On the other hand, since insignificant states can be rarely visited

Algorithm 2 Reward-free RL-Explore

1: **Input:** iteration number N_0, N.
2: set policy class $\Psi \leftarrow \emptyset$, and dataset $\mathcal{D} \leftarrow \emptyset$.
3: **for** all $(s,h) \in \mathcal{S} \times [H]$ **do**
4: $r_{h'}(s',a') \leftarrow \mathbb{1}[s' = s \text{ and } h' = h]$ for all $(s',a',h') \in \mathcal{S} \times \mathcal{A} \times [H]$.
5: $\Phi^{(s,h)} \leftarrow \text{EULER}(r, N_0)$.
6: $\pi_h(\cdot|s) \leftarrow \text{Uniform}(\mathcal{A})$ for all $\pi \in \Phi^{(s,h)}$.
7: $\Psi \leftarrow \Psi \cup \Phi^{(s,h)}$.
8: **end for**
9: **for** $n = 1 \dots N$ **do**
10: sample policy $\pi \sim \text{Uniform}(\Psi)$.
11: play $\mathcal{M}$ using policy π, and observe the trajectory $z_n = (s_1, a_1, \dots, s_H, a_H, s_{H+1})$.
12: $\mathcal{D} \leftarrow \mathcal{D} \cup \{z_n\}$
13: **end for**
14: **Return:** dataset $\mathcal{D}$.

no matter what policy is used, they will not significantly change the value from the initial states. Thus, for the sake of finding near-optimal policies, it is sufficient to visit all significant states with proper significance level ϵ. Indeed, Algorithm 2 is able to provide such a guarantee as follows.

Theorem 3.3. *There exists absolute constant $c > 0$ such that for any $\epsilon > 0$ and $p \in (0,1)$, if we set $N_0 \geq cS^2AH^4\iota_0^3/\delta$ where $\iota_0 := \log(SAH/(p\delta))$, then with probability at least $1-p$, that Algorithm 2 returns a dataset $\mathcal{D}$ consisting of N trajectories $\{z_n\}_{n=1}^N$, which are i.i.d sampled from a distribution μ satisfying:*

$$\forall\, \delta\text{-significant } (s,h), \quad \max_{a,\pi} \frac{P_h^\pi(s,a)}{\mu_h(s,a)} \leq 2SAH. \quad (4)$$

Theorem 3.3 claims that using Algorithm 2, we can collect data from a underlying distribution μ, which ensures that for policy π, the ratio $P_h^\pi(s,a)/\mu_h(s,a)$ will be upper bounded for any significant state and action. That is, all significant state and action will be visited by distribution μ with reasonable amount of probability. Notice as δ becomes smaller, there will be more significant states and the condition (4) becomes stronger. As a result we need to take larger N_0. As we will see later, the δ we take eventually will be $\epsilon/\left(2SH^2\right)$, where ϵ is the suboptimality of the policy we find in the planning phase.

Algorithm 2 can be decomposed into two parts, where Line 3-8 learns a set of exploration policies Ψ and Line 9-13 simply collects data by uniformly executing policies in Ψ. Therefore, the key mechanism lies in how to learn the set of exploration policies Ψ. Our strategy is to first learn the best policies that maximize the probability to research each state s at step h individually, and then combine them.

Concretely, for each state s at step h, algorithm 2 first creates a reward function r that is always zero except for the state s at step h. Then we can simulate a standard MDP by properly feeding this designed reward r when an agent interacts with the environment using protocol 1. It is easy to verify that the optimal policy for the MDP with this reward r is precisely the policy that maximizes the probability to reach (s,h). Thus, any RL algorithms with PAC or regret guarantees (Azar et al., 2017; Jin et al., 2018) can be used here to approximately find this optimal policy. In particular, we use EULER algorithm (Zanette & Brunskill, 2019), whose theoretical guarantee in our setting is presented as follows [2]

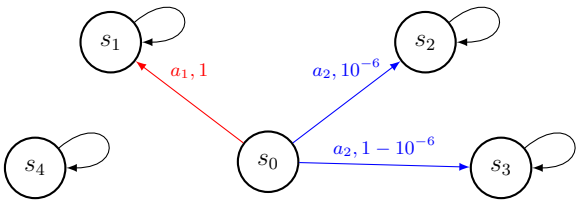

Figure 1: Illustration of significant states (Definition 3.2) v.s. insignificant states. In this toy example we have 5 states, where s_0 is the initial state. Only from state s_0 the agent can transit to other states and the other states are absorbing whatever action the agent takes. For state s_0, we use red arrows to represent transition if action a_1 is taken and blue ones if action a_2 is taken. The numbers on the arrows following the actions are the transition probability. In this example, s_4 is insignificant, because it can never be reached. For $\delta = 10^{-5}$, s_2 is also δ-insignificant, because the best policy to reach s_2 is by taking action a_2 at initial state s_0, which gives the maximum probability 10^{-6} to reach s_2. The remaining states s_1, s_3 are all δ-significant.

Lemma 3.4. *There exists absolute constant $c > 0$ such that for any $N_0 > 0$ and $p \in (0,1)$, with probability at least $1-p$, if we run* EULER *algorithm for N_0 episodes, it will output a policy set Φ with $|\Phi| = N_0$ that satisfies:*

$$\mathbb{E}_{s_1 \sim \mathbb{P}_1}\left[V_1^\star(s_1) - \frac{1}{N_0}\sum_{\pi \in \Phi} V_1^\pi(s_1)\right] \leq c \cdot \left\{\sqrt{\frac{SAH\iota_0 \cdot \mathbb{E}_{s_1 \sim \mathbb{P}_1} V_1^\star(s_1)}{N_0}} + \frac{S^2AH^4\iota_0^3}{N_0}\right\}$$

where $\iota_0 = \log(SAHN_0/p)$.

We comment that one unique feature of EULER algorithm is that its suboptimality scales with the value of the opti-

[2] In (Zanette & Brunskill, 2019), EULER is studied under stationary setting, where $\mathbb{P}$ and r does not depend on h. A stationary MDP can simulate a non-stationary MDP by augmenting state s to (s,h). Therefore, the effective number of states becomes SH when we apply the results in (Zanette & Brunskill, 2019).

Algorithm 3 Reward-free RL-Plan

1: **Input:** a dataset of transition $\mathcal{D}$, reward function r, accuracy ϵ.
2: **for** all $(s, a, s', h) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times [H]$ **do**
3: $N_h(s, a, s') \leftarrow \sum_{(s_h, a_h, s_{h+1}) \in \mathcal{D}} \mathbb{1}[s_h = s, a_h = a, s_{h+1} = s']$.
4: $N_h(s, a) \leftarrow \sum_{s'} N_h(s, a, s')$.
5: $\hat{\mathbb{P}}_h(s'|s, a) = N_h(s, a, s')/N_h(s, a)$.
6: **end for**
7: $\hat{\pi} \leftarrow$ APPROXIMATE-MDP-SOLVER$(\hat{\mathbb{P}}, r, \epsilon)$.
8: **Return:** policy $\hat{\pi}$.

mal policy $\mathbb{E}_{s_1 \sim \mathbb{P}_1} V_1^\star(s_1)$. This is key in obtaining a sharp result, and is especially helpful in dealing with those states that are still significant but their maximum reaching probability is low. Finally, since the best policy to reach (s, h) is only meaningful at steps before h, algorithm 2 then alters the policy for state s at step h to be Uniform$(\mathcal{A})$ to ensure good probability of choosing all actions for this state.

3.2. Planning Phase

In planning phase, the agent is given the reward function r, and aims to find a near-optimal policy based on r and dataset $\mathcal{D}$ collected in the exploration phase. Algorithm 3 proceeds with two steps. Line 2-6 use counts based on dataset $\mathcal{D}$ to estimate the empirical transition matrix $\hat{\mathbb{P}}$. Then, algorithm 3 calls a approximate MDP solver. Subroutine APPROXIMATE-MDP-SOLVER$(\hat{\mathbb{P}}, r, \epsilon)$ can be any algorithm that finds ϵ-suboptimal policy $\hat{\pi}$ for MDP with known transition matrix and reward (they are $\hat{\mathbb{P}}$, r in this case). See Section 3.3 for examples of such approximate MDP solvers.

Now we are ready to state the guarantee for Algorithm 3, which asserts that as long as the number of data collected in the exploration phase is sufficiently large, the output policy $\hat{\pi}$ is not only a near-optimal policy for the estimated MDP with transition $\hat{\mathbb{P}}$, but also a near-optimal policy for the true MDP.

Theorem 3.5. *There exists absolute constant $c > 0$, for any $\epsilon > 0$, $p \in (0, 1)$, assume dataset $\mathcal{D}$ has N i.i.d. samples from distribution μ which satisfies Eq.(4) with $\delta = \epsilon / \left(2SH^2\right)$, and $N \geq cH^5S^2A\iota/\epsilon^2$, then with probability at least $1 - p$, for any reward function r simultaneously, the output policy $\hat{\pi}$ of Algorithm 3 is 3ϵ-suboptimal. That is:*

$$\mathbb{E}_{s_1 \sim \mathbb{P}_1}[V_1^\star(s_1; r) - V_1^{\hat{\pi}}(s_1; r)] \leq 3\epsilon$$

The mechanism behind Theorem 3.5 is that: by sampling sufficient number of exploring data, we ensure that the empirical transition $\hat{\mathbb{P}}$ and the true transition $\mathbb{P}$ are close so that the near-optimal policy for the estimated MDP with transition $\hat{\mathbb{P}}$ is also near optimal for the true MDP. We note that the closeness of $\hat{\mathbb{P}}$ and $\mathbb{P}$ can not be established in the usual sense of the TV-distance (or other distributional distance) between $\hat{\mathbb{P}}_h(\cdot|s, a)$ and $\mathbb{P}_h(\cdot|s, a)$ is small for any (s, a, h), due to the existence of insignificant states. The key observation is that, nevertheless, we can establish the closeness of $\hat{\mathbb{P}}$ and $\mathbb{P}$ in the sense that for any policy π, the value functions starting from initial states are close. That is, the difference in policy evaluations of two MDPs is small, which is summarized in the following lemma.

Lemma 3.6. *Under the preconditions of Theorem 3.5, with probability at least $1-p$, for any reward function r and any policy π, we have:*

$$|\mathbb{E}_{s_1 \sim \mathbb{P}_1}[\hat{V}_1^\pi(s_1; r) - V_1^\pi(s_1; r)]| \leq \epsilon \tag{5}$$

where $\hat{V}$ is the value function of MDP with the transition $\hat{\mathbb{P}}$.

The establishment of Lemma 3.6 is a natural consequence of the following: (1) the total contribution from all insignificant states is small; (2) $\hat{\mathbb{P}}$ is reasonably accurate for all significant states; and (3) a new sharp concentration inequality (see Lemma C.2 in Appendix). With Lemma 3.6, now we are ready to prove Theorem 3.5.

Proof of Theorem 3.5. We denote the optimal policy of MDP$(\mathbb{P}, r)$ and MDP$(\hat{\mathbb{P}}, r)$ by $\pi^\star$ and $\hat{\pi}^\star$ respectively. The theorem is a direct consequence of the following decomposition

$$\begin{aligned}
&\mathbb{E}_{s_1 \sim \mathbb{P}_1}\{V_1^{\pi^\star}(s_1; r) - V_1^{\hat{\pi}}(s_1; r)\} \\
\leq &\underbrace{|\mathbb{E}_{s_1 \sim \mathbb{P}_1}\{V_1^{\pi^\star}(s_1; r) - \hat{V}_1^{\pi^\star}(s_1; r)\}|}_{\text{Evaluation error 1}} \\
&+ \underbrace{\mathbb{E}_{s_1 \sim \mathbb{P}_1}\{\hat{V}_1^{\pi^\star}(s_1; r) - \hat{V}_1^{\hat{\pi}^\star}(s_1; r)\}}_{\leq 0 \text{ by definition}} \\
&+ \underbrace{\mathbb{E}_{s_1 \sim \mathbb{P}_1}\{\hat{V}_1^{\hat{\pi}^\star}(s_1; r) - \hat{V}_1^{\hat{\pi}}(s_1; r)\}}_{\text{Optimization error}} \\
&+ \underbrace{|\mathbb{E}_{s_1 \sim \mathbb{P}_1}\{\hat{V}_1^{\hat{\pi}}(s_1; r) - V_1^{\hat{\pi}}(s_1; r)\}|}_{\text{Evaluation error 2}}
\end{aligned}$$

where evaluation errors are bounded by ϵ by Lemma 3.6 and optimization error is bounded by ϵ by assumption. □

3.3. Approximate MDP Solvers

Approximate MDP solvers aim to find a near-optimal policy when the exact transition matrix $\mathbb{P}$ and reward r are known. The simplest way to achieve this is by **Value Iteration** (VI) algorithm, which solves the Bellman optimality equation Eq.(2) in a dynamical programming fashion. Then

Algorithm 4 Natural Policy Gradient (NPG)

Input: transition matrix $\mathbb{P}$, reward function r, stepsize η, iteration number T.
initialize $\pi_h^{(0)}(\cdot|s) \leftarrow \text{Uniform}(\mathcal{A})$ for all (s,h)
for $t = 0, \cdots, T-1$ **do**
  evaluate $Q_h^{\pi^{(t)}}(s,a)$ using Bellman equation Eq.(1) for all (s,a,h).
  update $\pi_h^{(t+1)}(a|s) \propto \pi_h^{(t)}(a|s) \cdot \exp(\eta Q_h^{\pi^{(t)}}(s,a))$ for all (s,a,h).
end for
Return: policy $\pi^{(T)}$.

the greedy policy induced by the result $Q^\star$ gives precisely the optimal policy without error.

Another popular approach frequently used in practice is the **Natural Policy Gradient** (NPG) algorithm as shown in Algorithm 4. In each iteration, the algorithm first evaluates the value of policy $\pi^{(t)}$ using Bellman equation Eq.(1). Then it updates the policy by first scale it with the exponential of learning η times value $Q^{\pi^{(t)}}$, and then performs a normalization. For completeness, we provides its guarantee here, which resembles the infinite horizon analysis in (Agarwal et al., 2019)[3].

Proposition 3.7. *for any learning rate η and iteration number T, the output policy $\pi^{(T)}$ of Algorithm 4 satisfies the following:*

$$\mathbb{E}_{s_1\sim\mathbb{P}_1}[V_1^\star(s_1) - V_1^{\pi^{(T)}}(s_1)] \le \frac{H\log A}{\eta T} + \frac{H^2}{T}$$

Therefore, it is easy to verify, by choosing $\eta \ge \log A/H$ and $T = 2H^2/\epsilon$, the policy $\pi^{(T)}$ returned by NPG is ϵ-optimal. When $\eta \to \infty$, NPG reduces to policy iteration.

4. Lower Bound

In this section, we establish that $\Omega(H^2S^2A/\epsilon^2)$ trajectories are necessary to satisfy the guarantee from Theorem 3.1.

Theorem 4.1. *Let $C > 0$ be a universal constant. Then for $A \ge 2$, $S \ge C\log_2 A$, $H \ge C\log_2 S$, and any $\epsilon \le \min\{1/4, H/48\}$, any reward-free exploration algorithm* Alg *which statisfies the guarantee of Theorem 3.1 with $p = 1/2$ and accuracy parameter ϵ must collect $\Omega(S^2AH^2/\epsilon^2)$ trajectories in expectation. This is true even if* Alg *can return randomized or history-dependent (non-Markov) policies, and holds even if the rewards and transitions are identical across stages h.*

In particular, Theorem 4.1 shows that our upper bound (Theorem 3.1) is tight in S, A, ϵ, up to logarithmic factors and lower-order terms. Note that lower bound holds against querying an *unlimited* number of reward vectors. It is left as an open question whether such a lower bound holds when the algorithm is only required to ensure correctness over a smaller number of reward vectors pre-determined in advance. In what follows, we sketch a proof of Theorem 4.1; a formal proof is given in Appendix D.

[3]The guarantee here is slightly different from (Agarwal et al., 2019) because of difference choice of η. The η here is essential $\eta/(1-\gamma)$ in (Agarwal et al., 2019).

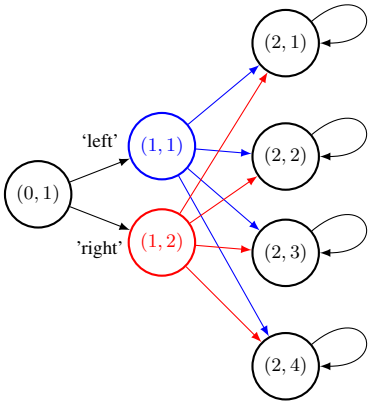

Figure 2: The "left" (blue) instance and "right" (red) instance embed two copies of the instance from Lemma 4.2. In each copy, the agent begins in stage $s = 0$, and moves to states $s \in [2n]$, $n = 2$. Different actions correspond to different probability distributions over next states $s \in [2n]$. States $s \in [2n]$ are absorbing, and rewards are action-independent. Lemma 4.2 shows that this construction requires the learner to learn $\Omega(n)$ bits about the transition probabilities $p(\cdot|0,a)$. By embedding this construction into a large MDP, this construction forces the learner to learn the transition probabilities at $n = 2$ states, $\{(x, \log_2 n) : x \in [n]\}$. The learner can determinstically access these states by appropriate choice of "left" and "right" actions.

4.1. Reward Free Exploration at a Single State

The core of our construction is a simple instance with a single initial state $x_1 = 0$ and $2n$ absorbing states $s \in [2n]$; the transition from states $0 \to s$ is described by a vector $q \in \mathbb{R}^{[2n]\times[A]}$, where $q(s,a)$ is the transition probability to state s if action a is taken at state 0. We shall also restrict to vectors q are close to uniform, i.e.,

$$\forall s, a, \quad \left| q(s,a) - \frac{1}{2n} \right| \le \frac{\epsilon}{2n} \tag{6}$$

The learner is then tasked with learning near optimal policies for reward vectors r_ν parametrized by $\nu \in [0,1]^{2n}$, which assigns a state-dependent but action-independent reward $\nu(s)$ to states $s \in [2n]$, and no reward to $x_1 = 0$.

The blue ("left") transitions or red ("right") transition in Figure 2 mirror this construction, which we formalize in Definition D.1. We show that reward-free exploration essentially forces the learner to learn the probability vectors $q(\cdot, a)$ in total-variation distance for each $a \in [A]$, yielding an $\Omega(nA/\epsilon^2)$ lower bound for this construction. A formal statement is of the following Lemma is given in Lemma D.2 in the appendix.

Lemma 4.2 (Informal)**.** *Suppose $S \geq C \log_2(A)$ for a universal constant $C > 0$. Suppose* Alg*, when faced with the instances described above (with q satisfying Eq.* (6)*) successfully returns ϵ-suboptimal policies for exponentially many reward vectors with total failure probability $1/2$. Then* Alg *requires $\Omega(SA/\epsilon^2)$ trajectories in expectation.*

Proof Sketch. Unfortunately, we cannot show a direct reduction from estimating q in total variation to learning near optimal-policies. Instead, by selecting appropriate reward vectors r_ν, the algorithm can decode a packing of $\exp(\Omega(n))$ transition vectors $q(\cdot, a)$ for each action $a \in [A]$. By a variant of Fano's inequality, this leads to the same $\Omega(nA/\epsilon^2)$ lower bound that would be obtained by a direct reduction. □

Lemma 4.2 differs from existing $\Omega(SA/\epsilon^2)$ lower bounds in that the only quantities unknown to the learner are the transition probabilities associated with the single state 0. This is in contrast to most existing lower bounds where the learner needs to collect transition information at multiple states. In particular, here the factor of S arises because the transition is *to* $\Theta(S)$ states, while in most constructions this factor arises because transitions *from* $\Theta(S)$ states must be estimated.

4.2. Lower Bound for Multiple States

To obtain an $\Omega(S^2H^2A/\epsilon^2)$ lower bound, we embed $n = \Omega(S)$ instances from above as the second-to-last layer of a binary tree of depth $1 + \log_2 n$. All n such instances share the same $2n$-terminal leaves (assume n is a power of 2). We index states by pairs (x, ℓ), where ℓ denotes the layer. From the binary tree construction, there are at most $4n$ states, so $n = \Omega(S)$. We assume that the MDP begins in stage $(0, 1)$, and for layers $\ell < \log_2 n$, action 1 always moves "left" in the tree, and actions $2, \ldots, A$ always moves "right" in the tree. Moreover, the leaf-states are all absorbing. The construction is given in Figure 2.

The only part unknown to the learner are the transition vectors $\{q_x\}_{x \in [n]}$, where $q_x(s, a)$ describes the probability of transitioning to leaf $(s, 1 + \log_2 n)$ when taking action a from state $(x, \log_2 n)$. We now index rewards by $(x, \nu) \in [n] \times [0, 1]^{2n}$, where $r_{x,\nu}$ places action-independent reward 1 on state $(x, \log_2 n)$, action-independent reward $\nu(s)$ on states $(s, 1 + \log_2 n)$, and reward 0 everywhere.

Assume that the transitions q_x satisfy the near-uniformity condition of (6) for $\epsilon = 1/4H$. Then, for reward $r_{x,\nu}$, the high reward of 1 at $(x, \log_2 n)$ forces any near-optimal policy to visit $(x, \log_2 n)$ and subsequently play near optimal actions at this state. However, playing optimally at $(x, \log_2 n)$ under reward $r_{x,\nu}$ for all ν is equivalent to reward-free learning of a single instance of the construction from Lemma 4.2. By varying $x \in [n]$ for the reward vectors $r_{x,\nu}$, the learner is forced to learn n such instances, yielding the $\Omega(n \cdot nA/\epsilon^2) = \Omega(S^2A/\epsilon^2)$ lower bound. This can be improved to $\Omega(H^2S^2A/\epsilon^2)$ by using the absorbing states to create a chain of $\Omega(H)$ rewards.

5. Conclusion

In this paper, we propose a new "reward-free RL" framework, comprising of two phases. In the exploration phase, the learner first collects trajectories from an MDP $\mathcal{M}$ *without* receiving any reward information. After the exploration phase, the learner is no longer allowed to interact with the MDP and she is instead tasked with computing near-optimal policies under for $\mathcal{M}$ for a collection of given reward functions. This framework is particularly suitable when there are many reward functions of interest, or when we are interested in learning the transition operator directly.

This paper provides an efficient algorithm that conducts $\widetilde{O}(S^2A\mathrm{poly}(H)/\epsilon^2)$ episodes of exploration and returns ϵ-suboptimal policies for an *arbitrary* number of adaptively chosen reward functions. Our planning procedure can be instantiated by any black-box approximate planner, such as value iteration or natural policy gradient. We also give a nearly-matching $\Omega(S^2AH^2/\epsilon^2)$ lower bound, demonstrating the near-optimality of our algorithm in this setting.

We close with some directions for future work. On the technical level, an interesting direction is to understand the sample complexity for reward-free RL with a pre-specified reward function that is unobserved during the exploration phase. Our lower bound proofs requires the agent to be able to optimize all possible reward functions, so it does not directly apply to this potentially easier setting. Can we use $\tilde{O}(SA\mathrm{poly}(H)/\epsilon^2)$ samples in the exploration phase to achieve this goal?

Another interesting direction is to design reward-free RL algorithms for settings with function approximation or infinite horizon. We believe our work highlights and introduces some mechanisms that may be useful in the these settings, such as the concept of significant states (Definition 3.2) and the coverage guarantee (4). How do we generalize these concepts?

We hope to pursue these directions in future work.

Acknowledgments

TY is partially supported by NSF BIGDATA grant IIS-1741341. MS is supported by an Open Philanthropy AI Fellowship, funded by the Good Ventures foundation.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31. JMLR. org, 2017.

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. Optimality and approximation with policy gradient methods in markov decision processes. *arXiv preprint arXiv:1908.00261*, 2019.

Altman, E. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Antos, A., Szepesvári, C., and Munos, R. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 263–272. JMLR. org, 2017.

Brafman, R. I. and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Cai, Q., Yang, Z., Jin, C., and Wang, Z. Provably efficient exploration in policy optimization. *arXiv preprint arXiv:1912.05830*, 2019.

Chen, J. and Jiang, N. Information-theoretic considerations in batch reinforcement learning. In *36th International Conference on Machine Learning, ICML 2019*, pp. 1792–1817. International Machine Learning Society (IMLS), 2019.

Chen, X., Guntuboyina, A., and Zhang, Y. On bayes risk lower bounds. *The Journal of Machine Learning Research*, 17(1):7687–7744, 2016.

Dann, C. and Brunskill, E. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2818–2826, 2015.

Dann, C., Lattimore, T., and Brunskill, E. Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 5713–5723, 2017.

Du, S., Krishnamurthy, A., Jiang, N., Agarwal, A., Dudik, M., and Langford, J. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, pp. 1665–1674, 2019.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *Proceedings of the 36th International Conference on Machine Learning*, number 97, 2019.

Jiang, N. UIUC CS598 statistical reinforcement learning lecture notes. 2019.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pp. 4863–4873, 2018.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

Kakade, S. M. *On the sample complexity of reinforcement learning*. PhD thesis, University of London, London, England, 2003.

Kaufmann, E., Cappé, O., and Garivier, A. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17 (1):1–42, 2016.

Kearns, M. and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

Li, L. Sample complexity bounds of exploration. In *Reinforcement Learning*, pp. 175–204. Springer, 2012.

Lim, S. H. and Auer, P. Autonomous exploration for navigating in mdps. In *Conference on Learning Theory*, pp. 40–1, 2012.

Miryoosefi, S., Brantley, K., Daume III, H., Dudik, M., and Schapire, R. E. Reinforcement learning with convex constraints. In *Advances in Neural Information Processing Systems*, pp. 14093–14102, 2019.

Misra, D., Henaff, M., Krishnamurthy, A., and Langford, J. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. *arXiv preprint arXiv:1911.05815*, 2019.

Munos, R. and Szepesvári, C. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

Simchowitz, M. and Jamieson, K. G. Non-asymptotic gap-dependent regret bounds for tabular mdps. In *Advances in Neural Information Processing Systems*, pp. 1151–1160, 2019.

Tarbouriech, J. and Lazaric, A. Active exploration in markov decision processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 974–982, 2019.

Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2018.

Zanette, A. and Brunskill, E. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, pp. 7304–7312, 2019.