Abstract

Suppose an agent is in a (possibly unknown) Markov Decision Process in the absence of a reward signal, what might we hope that an agent can efficiently learn to do? This work studies a broad class of objectives that are defined solely as functions of the state-visitation frequencies that are induced by how the agent behaves. For example, one natural, intrinsically defined, objective problem is for the agent to learn a policy which induces a distribution over state space that is as uniform as possible, which can be measured in an entropic sense. We provide an efficient algorithm to optimize such intrinsically defined objectives, when given access to a black box planning oracle (which is robust to function approximation). Furthermore, when restricted to the tabular setting where we have sample based access to the MDP, our proposed algorithm is provably efficient, both in terms of its sample and computational complexities. Key to our algorithmic methodology is utilizing the conditional gradient method (a.k.a. the Frank-Wolfe algorithm) which utilizes an approximate MDP solver.

1 Introduction

A fundamental problem in reinforcement learning is that of exploring the state space. How do we understand what is even possible in a given environment without a reward signal? This question has received a lot of attention, with approaches such as learning with intrinsic reward and curiosity driven methods, surveyed below. Our work studies a class of objectives that is defined solely as function of the state-visitation frequencies. A natural such objective is finding a policy that maximizes the entropy of the induced distribution over the state space. More generally, our approach extends to any concave function over distributions.

Suppose the MDP is fully and precisely known, in terms of states, actions, and the entire transition matrix. Then maximizing the entropy can be recast as a convex optimization problem (see Section 3.2.2 or [DFVR03]) over the space of state-visitation frequencies induced by the exhaustive set of all policies. However, most RL instances that are common in practice exhibit at least one of several complications:

— prohibitively large state space (i.e. Chess or Go)
— unknown transition matrix (as in common Atari games)

These scenarios often require function approximation, i.e. restricting the search to a non-linearly parameterized policy class (e.g. neural networks), which makes the entropy maximization problem non-convex.

As a remedy for the computational difficulty, we propose considering an approximate planning oracle: an efficient method that given a well-specified reward signal can find an optimizing policy. Such sample-based planning oracles have been empirically observed to work well with non-linear policy classes. Given such an oracle, we give a provably efficient method for exploration based on the conditional gradient (or Frank-Wolfe) algorithm [FW56].
Formally, we show how to generate a sequence of reward signals, that sequentially optimized give rise to a policy with entropy close to optimal. Our main theorem gives a bound on the number of calls to the planning oracle, which is independent of the size of the state space of the MDP. Next, we outline the efficient construction of such oracles and state the resultant sample & computational complexity in the tabular MDP setting. As a proof of concept, we implement our method and show experiments over several mainstream RL tasks in Section 5.

1.1 Informal statement of contributions

To facilitate exploration in potentially unknown MDPs within a restricted policy class, we assume access to the environment using the following two oracles:

**Approximate planning oracle:** Given a reward function (on states) \( r : S \rightarrow \mathbb{R} \) and a sub-optimality gap \( \varepsilon \), the planning oracle returns a stationary policy \( \pi = \text{APPROXPLAN}(r, \varepsilon) \) with the guarantee that \( V(\pi) \geq \max_\pi V(\pi) - \varepsilon \), where \( V(\pi) \) is the value of policy \( \pi \).

**State distribution estimate oracle:** A state distribution oracle estimates the state distribution \( \hat{d}_{\pi} = \text{DENSITYEST}(\pi, \varepsilon) \) of any given (non-stationary) policy \( \pi \), guaranteeing that \( \|d_{\pi} - \hat{d}_{\pi}\|_\infty \leq \varepsilon \).

Given access to these two oracles, we describe a method that provably optimizes any continuous and smooth objective over the state-visitation frequencies. Of special interest is the maximum entropy and relative entropy objectives.

**Theorem 1.1** (Main Theorem - Informal). There exists an efficient algorithm (Algorithm 1) such that for any \( \beta \)-smooth measure \( R \), and any \( \varepsilon > 0 \), in \( O\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right) \) calls to \text{APPROXPLAN} & \text{DENSITYEST} , it returns a policy \( \bar{\pi} \) with \( R(d_{\bar{\pi}}) \geq \max_\pi R(d_{\pi}) - \varepsilon \).

1.2 Related work

We review related works in this section.

**Reward Shaping & Imitation Learning:** Direct optimization approaches to RL (such as policy gradient methods) tend to perform favorably when random sequences of actions lead the agent to some positive reward, but tend to fail when the rewards are sparse or myopic. Thus far, the most practical approaches to address this have either been through some carefully constructed reward shaping (e.g. [NHR99] where dense reward functions are provided to make the optimization problem more tractable) or through inverse reinforcement learning and imitation learning [AN04, RGB11] (where an expert demonstrates to the agent how to act).

**PAC Learning:** For the case of tabular Markov decision processes, the balance of exploration and exploitation has been addressed in that there are a number of methods which utilize confidence based reward bonuses to encourage exploration in order to ultimately behave near optimally [KS02, Kak03, SLW+06, LH14, DB15, SS10, AOM17].

**Count-based Models & Directed Exploration:** There are a host of recent empirical success using deep RL methods which encourage exploration in some form [MKS+15, SHM+16]. The approaches which encourage exploration are based on a few related ideas: that of encouraging encouraging exploration through state visitation frequencies (e.g. [OBvdOM17, BSQ+16, THF+17]) and those based on a intrinsic reward signal derived from novelty or prediction error [LLTyO12, PAED17, SRM+18, FCRL17, MJR15, HCC+16, WRR+17], aligning an intrinsic reward to the target objective [Kae93, CBS05, SLBS10, SLB09, ZOS18], or sample based approaches to tracking of value function uncertainty [OBPVR16, OAC18].

**Intrinsic Learning:** Works in [CBS05, SLB09, SLBS10] established computational theories of intrinsic reward signals (and how it might help with downstream learning of tasks) and other works also showed how to incorporate intrinsic rewards (in the absence of any true reward signal) [WdWK+18, BESK18, BPP+18, NPD+18]. The potential benefit is that such learning may help the agent reach a variety of achievable goals and do well on other extrinsically defined tasks, not just the task under which it was explicitly trained for under one specific reward function (e.g. see [CBS05, SLB09, WdWK+18, NPD+18]).
2 Preliminaries

Markov decision process: An infinite-horizon discounted Markov Decision Process is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \gamma, d_0)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $d_0$ is the distribution of of the initial state $s_0$. At each timestep $t$, upon observing the state $s_t$, the execution of action $a_t$ triggers an observable reward of $r_t = r(s_t, a_t)$ and a transition to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$. The performance on an infinite sequence of states & actions (hereafter, referred to as a trajectory) is judged through the (discounted) cumulative reward it accumulates, defined as

$$V(\tau = (s_0, a_0, s_1, a_1, \ldots )) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t).$$

Policies: A policy is a (randomized) mapping from a history, say $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots s_{t-1}, a_{t-1}, r_{t-1})$, to an action $a_t$. A stationary policy is a (randomized) function which maps a state to an action in a time-independent manner, i.e. $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$. When a policy $\pi$ is executed on some MDP $\mathcal{M}$, it produces a distribution over infinite-length trajectories $\tau = (s_0, a_0, s_1, a_1, \ldots )$ as specified below.

$$P(\tau|\pi) = P(s_0) \prod_{i=0}^{\infty} (\pi(a_i|s_i)P(s_{i+1}|s_i, a_i))$$

The (discounted) value $V_\pi$ of a policy $\pi$ is the expected cumulative reward an action sequence sampled from the policy $\pi$ gathers.

$$V_\pi = \mathbb{E}_{\tau \sim P(\cdot|\pi)} V(\tau) = (1 - \gamma) \mathbb{E}_{\tau \sim P(\cdot|\pi)} \sum_{i=0}^{\infty} \gamma^i r(s_i, a_i)$$

Induced state distributions: The $t$-step state distribution and the (discounted) state distribution of a policy $\pi$ that result are

$$d_{t,\pi}(s) = P(s_t = s|\pi) = \sum_{\text{all } \tau \text{ with } s_t = s} P(\tau|\pi), \quad (2.1)$$

$$d_{t,\pi}(s,a) = P(s_t = s, a_t = a|\pi) = \sum_{\text{all } \tau \text{ with } s_t = s, a_t = a} P(\tau|\pi), \quad (2.2)$$

$$d_\pi(s) = (1 - \gamma) \sum_{t=1}^{\infty} \gamma^t d_{t,\pi}(s), \quad (2.3)$$

$$d_\pi(s,a) = (1 - \gamma) \sum_{t=1}^{\infty} \gamma^t d_{t,\pi}(s,a). \quad (2.4)$$

The latter distribution can be viewed as the analogue of the stationary distribution in the infinite horizon setting.

Mixtures of stationary policies: Given a sequence of $k$ policies $C = (\pi_0, \ldots \pi_{k-1})$, and $\alpha \in \Delta_k$, we define $\pi_{\text{mix}} = (\alpha, C)$ to be a mixture over stationary policies. The (non-stationary) policy $\pi_{\text{mix}}$ is one where, at the first timestep $t = 0$, we sample policy $\pi_i$ with probability $a_i$ and then use this policy for all subsequent timesteps. In particular, the behavior of a mixture $\pi_{\text{mix}}$ with respect to an MDP is that it induces infinite-length trajectories $\tau = (s_0, a_0, s_1, a_1 \ldots )$ with the probability law :

$$P(\tau|\pi_{\text{mix}}) = \sum_{i=0}^{k-1} \alpha_i P(\tau|\pi_i) \quad (2.5)$$

and the induced state distribution is:

$$d_{\pi_{\text{mix}}}(s) = \sum_{i=0}^{k-1} \alpha_i d_{\pi_i}(s). \quad (2.6)$$
Note that such a distribution over policies need not be representable as a stationary stochastic policy (even if the \( \pi_i \)'s are stationary) due to that the sampled actions are no longer conditionally independent given the states.

3 The Objective: MaxEnt Exploration

As each policy induces a distribution over states, we can associate a \textit{concave} reward functional \( R(\cdot) \) with this induced distribution. We say that a policy \( \pi^* \) is a \textit{maximum-entropy exploration policy}, also to referred to as the \textit{max-ent} policy, if the corresponding induced state distribution has the maximum possible \( R(d_\pi) \) among the class of all policies. Lemma 3.3 assures us that the search over the class of stationary policies is sufficient.

\[ \pi^* \in \arg \max_\pi R(d_\pi). \]

Our goal is to find a policy that induces a state distribution with a comparable value of the reward functional.

3.1 Examples of reward functionals

A possible quantity of interest that serves as a motivation for considering such functionals is the entropy of the induced distribution.

\[ \max_\pi \{ H(d_\pi) = - \mathbb{E}_{s \sim d_\pi} \log d_\pi(s) \} \]

The same techniques we derive can also be used to optimize other entropic measures. For example, we may be interested in minimizing:

\[ \min_\pi \left\{ \text{KL}(d_\pi||Q) = \mathbb{E}_{s \sim d_\pi} \log \frac{d_\pi(s)}{Q(s)} \right\} \]

for some given distribution \( Q(s) \). Alternatively, we may seek to minimize a cross entropy measure:

\[ \min_\pi \left\{ \mathbb{E}_{s \sim Q} \log \frac{1}{d_\pi(s)} = \text{KL}(Q||d_\pi) + H(Q) \right\} \]

where the expectation is now under \( Q \). For uniform \( Q \), this latter measure may be more aggressive in forcing \( \pi \) to have more uniform coverage than the entropy objective.

3.2 Landscape of the objective function

In this section, we establish that the entropy of the state distribution is \textit{not} a concave function of the policy. Similar constructions can establish analogous statements for other non-trivial functionals. Subsequently, we discuss a possible convex reformulation of the objective in the space of induced distributions which constitute a convex set.

3.2.1 Non-convexity in the policy space

Despite the concavity of the entropy function, our overall maximization problem is not concave as the state distribution is not an affine function of the policy. This is stated precisely in the following lemma.

\textbf{Lemma 3.1.} \( H(d_\pi) \) is \textit{not} concave in \( \pi \).  

\textit{Proof.} Figure 1 demonstrates the behavior of \( \pi_0, \pi_1, \pi_2 \) on a 6-state MDP with binary actions. Note that for sufficiently large \( \gamma \to 1 \) and any policy \( \pi \), the discounted state distribution converges to the distribution on the states at the second timestep, or formally \( d_\pi \to d_{2,\pi} \). Now with the realization \( \pi_0 = \frac{2\pi_1 + \pi_2}{2} \), observe that \( d_{2,\pi_0} \) is not uniform on \( \{s_{2,0}, s_{2,1}, s_{2,2}\} \), implying that \( H(d_{2,\pi_0}) < \frac{H(d_{2,\pi_1}) + H(d_{2,\pi_2})}{2} \). \( \square \)
Lemma 3.2. For any policy \( \pi \) and MDP \( M \), define the matrix \( P_\pi \in \mathbb{R}^{|S| \times |S|} \) so that
\[
P_\pi(s', s) = \sum_{a \in A} \pi(a|s)P(s'|s, a).
\]
Then it is true that
1. \( P_\pi \) is linear in \( \pi \),
2. \( d_{t, \pi} = P_t d_0 \) for all \( t \geq 0 \),
3. \( d_\pi = (1 - \gamma)(I - \gamma P_\pi)^{-1}d_0 \).

**Proof.** Linearity of \( P_\pi \) is evident from the definition. (2,3) may be verified by calculation.

3.2.2 Convexity in the distribution space

Define the set of all induced distributions as \( K = \{ d : d(s, a) \geq 0 \text{ and satisfies the constraints stated below} \} \). For every \( d \in K \), it is possible to construct a policy \( \pi \) with \( d_\pi = d \), and for every \( \pi \), \( d_\pi \in K \) holds [Put14].

\[
\sum_a d(s, a) = (1 - \gamma)d_0(s) + \gamma \sum_{s', a'} P(s|s', a')d(s', a')
\]

The search for a max-ent policy can be recast as a convex optimization problem over the space of distributions.

\[
\max_{d \in K} R(d).
\]

Although we outline this reduction for an unrestricted policy class, similar reductions are possible for linearly-parameterized policy classes. These techniques can be extended to the case of MDPs with unknown dynamics [DFVR03].

3.2.3 Sufficiency of Stationary Policies

The set of non-Markovian policies is *richer* than the set of Markov stationary policies in terms of the distributions over trajectories each may induce. A priori, it is not evident that maximizing \( R(d_\pi) \) over the set of stationary policies is sufficient to guarantee the optimality in a larger class of all policies. Lemma 3.3 establishes this claim by equating the set of achievable *induced state distributions* for these two sets of policies.

**Lemma 3.3.** [Put14] For any possibly non-Markovian policy \( \pi \), define a stationary Markov policy \( \pi' \) as \( \pi'(a|s) = \frac{d_\pi(s, a)}{d_\pi(s)} \). Then, \( d_\pi = d_{\pi'} \).
Algorithm 1 Maximum-entropy policy computation.

1: **Input:** Step size $\eta$, number of iterations $T$, planning oracle error tolerance $\varepsilon_1 > 0$, state distribution oracle error tolerance $\varepsilon_0 > 0$, reward functional $R$.
2: Set $C_0 = \{\pi_0\}$ where $\pi_0$ is an arbitrary policy.
3: Set $\alpha_0 = 1$.
4: for $t = 0, \ldots, T - 1$ do
5: Call the state distribution oracle on $\pi_{\text{mix},t} = (\alpha_t, C_t)$:
   \[ \hat{d}_{\pi_{\text{mix},t}} = \text{DensityEst}(\pi_{\text{mix},t}, \varepsilon_0) \]
6: Define the reward function $r_t$ as
   \[ r_t(s) = \nabla R(\hat{d}_{\pi_{\text{mix},t}}) := \left. \frac{dR(X)}{dX} \right|_{X=\hat{d}_{\pi_{\text{mix},t}}} \]
7: Compute the (approximately) optimal policy on $r_t$:
   \[ \pi_{t+1} = \text{ApproxPlan}(r_t, \varepsilon_1) \]
8: Update $\pi_{\text{mix},t+1} = (\alpha_{t+1}, C_{t+1})$ to be
   \[ C_{t+1} = (\pi_0, \ldots, \pi_t, \pi_{t+1}), \]
   \[ \alpha_{t+1} = ((1 - \eta)\alpha_t, \eta). \] (4.1) (4.2)
9: end for
10: return $\pi_{\text{mix},T} = (\alpha_T, C_T)$.

4 Algorithms & Main Results

The algorithm maintains a distribution over policies, and proceeds by adding a new policy to the support of the mixture and reweighing the components. To describe the algorithm, we will utilize access to two kinds of oracles. The constructions for these are detailed in later sections.

**Approximate planning oracle:** Given a reward function (on states) $r : S \to \mathbb{R}$ and a sub-optimality gap $\varepsilon_1$, the planning oracle returns a stationary policy $\pi = \text{ApproxPlan}(r, \varepsilon_1)$ with the guarantee that $V_\pi \geq \max_{\pi} V_\pi - \varepsilon_1$.

**State distribution estimate oracle:** A state distribution oracle estimates the state distribution $\hat{d}_\pi = \text{DensityEst}(\pi, \varepsilon_0)$ of any given (non-stationary) policy $\pi$, guaranteeing that $\|d_\pi - \hat{d}_\pi\|_\infty \leq \varepsilon_0$.

We shall assume in the following discussion that the reward functional $R$ is $\beta$-smooth, $B$-bounded, and that it satisfies the following inequality for all $X,Y$.

\[ \|\nabla R(X) - \nabla R(Y)\|_\infty \leq \beta \|X - Y\|_\infty \]
\[ - \beta \leq \nabla^2 R(X) \leq \beta \mathbf{I}; \quad \|\nabla R(X)\|_\infty \leq B \] (4.3) (4.4)

**Theorem 4.1** (Main Theorem). For any $\varepsilon > 0$, set $\varepsilon_1 = 0.1\varepsilon$, $\varepsilon_0 = 0.1\beta^{-1}\varepsilon$, and $\eta = 0.1\beta^{-1}\varepsilon$. When Algorithm 1 is run for $T$ iterations where:

\[ T \geq 10\beta\varepsilon^{-1} \log 10Be^{-1}, \]

we have that:

\[ R(\pi_{\text{mix},T}) \geq \max_{\pi} R(d_\pi) - \varepsilon. \]

\[ ^1 \text{As the oracle is solving a discounted problem, we know the optimal value is achieved by a stationary policy.} \]
Before we begin the proof, we state the implication for maximizing the entropy of the induced distribution. While the entropy objective is, strictly speaking, not smooth, one may consider a smoothed alternative $H_\sigma$ defined below.

$$H_\sigma(d_\pi) = -E_{s \sim d_\pi} \log(d_\pi(s) + \sigma)$$

When the algorithm is fed $H_\sigma$ as the proxy reward functional, it is possible make sub-optimality guarantees on the true objective $H$. Lemma 4.3(D) relates the entropy functional $H$ to its smoothed variant $H_\sigma$, while the rest of the lemma quantifies smoothness of $H_\sigma$. The factors of $|S|$ incurred below are a consequence of imposed smoothing on $H$, and are not necessary for naturally smooth objectives.

**Corollary 4.2.** For any $\varepsilon > 0$, set $\sigma = \frac{0.1\varepsilon}{2|S|}$, $\varepsilon_1 = 0.1\varepsilon$, $\varepsilon_0 = \frac{0.1\varepsilon^2}{80|S|}$, and $\eta = \frac{0.1\varepsilon^2}{40|S|}$. When Algorithm 1 is run for $T$ iterations with the reward functional $H_\sigma$, where:

$$T \geq \frac{40|S|}{0.1\varepsilon^2} \log \frac{|S|}{0.1\varepsilon^2},$$

we have that:

$$H(\pi_{\text{mix},T}) \geq \max_{\pi} H(d_\pi) - \varepsilon.$$
The following lemma is helpful in proving the corollary for the entropy functional.

**Lemma 4.3.** For any two distributions \( P, Q \in \Delta_d \):

(A) \((\nabla H_\sigma(P))_i = - \left( \log(P_i + \sigma) + \frac{P_i}{P_i + \sigma} \right)\)

(B) \( H_\sigma(P) \) is concave in \( P \),

(C) \( H_\sigma(P) \) is \( 2\sigma^{-1} \) smooth, ie. 

\[-2\sigma^{-1}I_d \preceq \nabla^2 R(P) \preceq 2\sigma^{-1}I_d,\]

(D) \(|H_\sigma(P) - R(P)| \leq d\sigma,\)

(E) \( \|\nabla H_\sigma(P) - \nabla H_\sigma(Q)\|_\infty \leq 2\sigma^{-1}\|P - Q\|_\infty.\)

**Proof of Lemma 4.3.** (A) may be verified by explicit calculation. Observe \( (\nabla^2 H_\sigma(P))_{i,i} \) is a diagonal matrix with entries

\[(\nabla^2 H_\sigma(P))_{i,i} = - \frac{P_i + 2\sigma}{(P_i + \sigma)^2}.\]

(B) is immediate. (C) follows as \( |(\nabla^2 H_\sigma(P))_{i,i}| \leq 2\sigma^{-1}.\)

\[|H_\sigma(P) - H(P)| = \sum_{i=0}^{d-1} P_i \log \frac{P_i + \sigma}{P_i} \leq \sum_{i=0}^{d-1} P_i \frac{\sigma}{P_i} = d\sigma.\]

The last inequality follows from \( \log x \leq x - 1, \forall x > 0. \) Finally, to see (E), using Taylor’s theorem, observe

\[\|\nabla H_\sigma(P) - \nabla H_\sigma(Q)\|_\infty \leq \max_{i,\alpha \in [0,1]} |(\nabla^2 H_\sigma(\alpha P + (1-\alpha)Q))_{i,i}|\|P - Q\|_\infty \]

\[\leq 2\sigma^{-1}\|P - Q\|_\infty.\]

\[\square\]

### 4.1 Tabular setting

In general, the construction of provably computationally efficient approximate planning oracle for MDPs with large or continuous state spaces poses a challenge. Discounting limited settings (eg. the Linear Quadratic Regulators [Ber05], [FGKM18]), one may only appeal to the recent empirical successes of sample-based planning algorithms that rely on the power of non-linear function approximation.

Nevertheless, one may expect, and possibly require, that any solution proposed to address the general case performs reasonably when restricted to the tabular setting. In this spirit, we outline the construction of the required oracles in the tabular setting.

#### 4.1.1 The known MDP case

With the knowledge of the transition matrix \( P \) of a MDP \( \mathcal{M} \) in the form of an explicit tensor, the planning oracle can be implemented via any of the exact solution methods [Ber05], eg. value iteration, linear programming. The state distribution oracle can be efficiently implemented as Lemma 3.2 suggests.

**Corollary 4.4.** When the MDP \( \mathcal{M} \) is known explicitly, with the oracles described in Section 4, Algorithm 7 runs in \( \text{poly} \left( \beta, |S|, |\mathcal{A}|, \frac{1}{1-\gamma}, \frac{1}{\varepsilon}, \log B \right) \) time to guarantee \( R(d_{\pi_{\text{mix}}}, T) \geq \max_\pi R(d_\pi) - \varepsilon. \)
4.1.2 The unknown MDP case

For the case of an unknown MDP, a sample-based algorithm must iteratively try to learn about the MDP through its interactions with the environment. Here, we assume a $\gamma$-discounted episodic setting, where the agent can act in the environment starting from $s_0 \sim d_0$ for some number of steps, and is then able to reset. Our measure of sample complexity in this setting is the number of $\tilde{O}((1 - \gamma)^{-1})$-length episodes the agent must sample to achieve a $\varepsilon$-suboptimal performance guarantee.

The algorithm outlined below makes a distinction between the set of states it is (relatively) sure about and the set of states that have not been visited enough number of times yet. The algorithm and the analysis is similar to the $E^3$ algorithm [KS02]. Since algorithms like $E^3$ proceed by building a relatively accurate model on the set of reachable states, as opposed to estimate of the value functions, this permits the reuse of information across different invocations, each of which might operate on a different reward signal.

Theorem 4.5. For an unknown MDP, with Algorithm 3 as the planning oracle and Algorithm 3 as the distribution estimate oracle, Algorithm 1 runs in $\tilde{O}\left(\frac{\log |S| \varepsilon^{-1}}{\log \gamma^{-1}}\right)$ time and executes $\tilde{O}\left(\frac{\log |S| \varepsilon^{-1}}{\log \gamma^{-1}} + \frac{\log |A|}{\varepsilon}\right)$ episodes of length $\tilde{O}\left(\frac{1}{\log \gamma^{-1}}\right)$ to guarantee that

$$R(d_{\pi_{\text{mix}}}, T) \geq \max_{\pi} R(d_{\pi}) - \varepsilon.$$ 

A sub-optimality bound may be derived on the non-smooth entropy functional $H$ via Lemma 4.3. Again, the extraneous factors introduced in the process are a consequence of the imposed smoothing via $H_\sigma$.

Corollary 4.6. For an unknown MDP, with Algorithm 3 as the planning oracle and Algorithm 3 as the distribution estimate oracle and $H_\sigma$ as the proxy reward functional, Algorithm 1 runs in $\tilde{O}\left(|S|, |A|, \frac{1}{1 - \gamma}, \frac{1}{\varepsilon}\right)$ time and executes $\tilde{O}\left(\frac{|S|^2|A|}{\varepsilon^3(1 - \gamma)^2} + \frac{|S|}{\varepsilon^6}\right)$ episodes of length $\tilde{O}\left(\frac{1}{\log \gamma^{-1}}\right)$ to guarantee that

$$H(d_{\pi_{\text{mix}}}, T) \geq \max_{\pi} H(d_{\pi}) - \varepsilon.$$ 

Before we state the proof, we note the following lemmas. The first is an adaptation of the analysis of the $E^3$ algorithm. The second is standard. We only include the second for completeness. The proofs of these may be found in the appendix.

Lemma 4.7. For any reward function $r$ with $\|r\|_\infty \leq B$, $\varepsilon > 0$, with $\varepsilon_1 = 0.1B^{-1}\varepsilon$, $m = \frac{32|S|^2|A| \log \frac{2|S|}{\varepsilon_1}}{0.1B(1 - \gamma)^2}$, $n = \frac{B \log \frac{32|S|^2|A| \log \frac{2|S|}{\varepsilon_1}}{0.1B(1 - \gamma)^2}}{(1 - \gamma)^2(0.1B)^2}$, $t_0 = \frac{\log \frac{0.1B}{\varepsilon_1}}{\log \gamma}$, Algorithm 4.5 guarantees with probability $1 - \delta$

$$V_\pi \geq \max_{\pi} V_\pi - \varepsilon.$$

Furthermore, note that if Algorithm 4.5 is invoked $T$ times (on possibly different reward functions), the total number of episodes sampled across all the invocations is $n(T + m|S||A|) = \tilde{O}\left(\frac{B'T}{\varepsilon} + \frac{B^3|S|^2|A|}{\varepsilon^3(1 - \gamma)^2}\right)$, each episode being of length $t_0$.

Lemma 4.8. For any $\varepsilon_0, \delta > 0$, when Algorithm 3 is run with $m = \frac{200|S| \log \frac{2|S| \log 0.1B}{\varepsilon_0}}{0.1B}$, $t_0 = \frac{\log 0.1B}{\log \gamma}$, $\hat{d}_\pi$ satisfies $\|\hat{d}_\pi - d_\pi\|_\infty \leq \varepsilon_0$ with probability at least $1 - \delta$. In this process, the algorithm samples $m$ episodes of length $t_0$.

Proof of Theorem 4.5. The claim follows immediately from the invocations of the two lemmas above with the parameter settings proposed in Theorem 4.1. 

The following notions & lemmas are helpful in proving Lemma 4.7. We shall call a state $s \in K$ $m$-known if, for all actions $a \in A$, action $a$ has been executed at state $s$ at least $m$ times. For any MDP $\mathcal{M} = (S, A, r, P, \gamma)$
Algorithm 2 Sample-based planning for an unknown MDP.

1: **Input:** Reward \( r \), error tolerance \( \varepsilon > 0 \), exact planning oracle tolerance \( \varepsilon_1 > 0 \), oversampling parameter \( m \), number of rollouts \( n \), rollout length \( t_0 \).

2: Initialize a persistent data structure \( C \in \mathbb{R}^{\mid S \mid^2 \times \mid A \mid} \), which is maintained across different calls to the planning algorithm to keep transition counts, to \( C(s'|s,a) = 0 \) for every \((s',s,a) \in S^2 \times A\).

3: \textbf{repeat}

4: Declare \( K = \{ s : \min_{a \in A} \sum_{s' \in S} C(s'|s,a) \geq m \} \), \( \hat{P}(s'|s,a) = \begin{cases} \frac{C(s'|s,a)}{\sum_{s' \in S} C(s'|s,a)} & \text{if } s \in K \\ 0 & \text{otherwise} \end{cases} \).

5: Define the reward function as \( r_K(s) = \begin{cases} r(s) & \text{if } s \in K \\ B & \text{otherwise} \end{cases} \).

6: Compute an (approximately) optimal policy on the MDP induced by \( \hat{P} \) and reward \( r_K \). This task is purely computational, and can be done as indicated in Section 4.1.1. Also, modify the policy so that on every state \( s \in S - K \), it chooses the least performed action.

\[ \pi(s) = \begin{cases} (\Pi (r_K, \varepsilon_1))(s) & \text{if } s \in K \\ \arg\min_{a \in A} \sum_{s' \in S} C(s'|s,a) & \text{otherwise} \end{cases} \]

7: Run \( \pi \) on the true MDP \( M \) to obtain \( n \) independently sampled \( t_0 \)-length trajectories \((\tau_1, \ldots \tau_n)\), and increment the corresponding counts in \( C(s'|s,a) \).

8: If and only if no trajectory \( \tau_i \) contains a state \( s \in S - K \), mark \( \pi \) as \textit{stable}.

9: \textbf{until} \( \pi \) is stable.

10: \textbf{return} \( \pi \).

Algorithm 3 Sample-based estimate of the state distribution.

1: **Input:** A policy \( \pi \), termination length \( t_0 \), oversampling parameter \( m \).

2: Sample \( m \) trajectories \((\tau_0, \ldots \tau_{m-1})\) of length \( t_0 \) following the policy \( \pi \).

3: For every \( t < t_0 \), calculate the empirical state distribution \( \hat{d}_{t,\pi} \).

\[ \hat{d}_{t,\pi}(s) = \frac{|\{i < m : \tau_i = (s_0, a_0, \ldots) \text{ with } s_t = s\}|}{m} \]

4: \textbf{return} \( \hat{d}_\pi = \frac{1 - \gamma}{1 - \gamma^{t_0}} \sum_{t=0}^{t_0-1} \gamma^t \hat{d}_{t,\pi} \)

and a set of \( m \)-known states \( K \subseteq S \), define an induced MDP \( M_K = (S, A, r_K, P_K, \gamma) \) so that the states absent from \( K \) are absorbing and maximally rewarding.

\[ r_K(s,a) = \begin{cases} r(s,a) & \text{if } s \in K \\ B & \text{otherwise} \end{cases} \] \hspace{1cm} (4.5)

\[ P_K(s'|s,a) = \begin{cases} P(s'|s,a) & \text{if } s \in K \\ 1_{s'=s} & \text{otherwise} \end{cases} \] \hspace{1cm} (4.6)

The state distribution induced by a policy \( \pi \) on \( M_K \) shall be denoted by \( d_{M_K,\pi} \). Often, in absence of an exact knowledge of the transition matrix \( P \), the policy \( \pi \) may be executed on an estimated transition matrix \( \hat{P} \). We shall use \( d_{M_K,\hat{P},\pi} \) to denote the state distribution of the policy \( \pi \) executed on the MDP with the transition
matrix $\hat{P}$. Also, define the following.

$$P_K(\text{escape}|\pi) = \sum_{s \in S} \sum_{a \in A} \sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{s_t \in K \forall a < t}$$

Note that $P_K(\text{escape}|\pi) \geq P_{K,\gamma}(\text{escape}|\pi) - \gamma^t \delta$.

**Lemma 4.9.** (Lemma 8.4.4 [Kak03]) For any policy $\pi$, the following statements are valid.

$$\langle d_\pi, r \rangle \geq \langle d_{M_{K,\pi}}, r_K \rangle - P_{K,\gamma}(\text{escape}|\pi)\|r_K\|_\infty$$

**Lemma 4.10.** ( Lemma 8.5.4 [Kak03]) If, for all $(s,a) \in S \times A$, $\|\hat{P}(\cdot|s,a) - P_K(\cdot|s,a)\|_1 \leq \varepsilon$, then for any reward $r$, policy $\pi$, it is true that

$$|\langle d_{M_{K,\pi}}, r \rangle - \langle d_{\hat{M}_{K,\pi}}, r \rangle| \leq \frac{\varepsilon}{1 - \gamma}$$

**Lemma 4.11.** (Folklore, eg. Lemma 8.5.5 [Kak03]) When $m$ samples $\{x_1, \ldots, x_m\}$ are drawn from a distribution $P$, supported on a domain of size $d$, to construct an empirical distribution $\hat{P}(\cdot|x) = \sum_{i=1}^{m} \mathbb{1}_{x_i = x}$, it is guaranteed that with probability $1 - \delta$

$$\|P - \hat{P}\|_1 \leq \sqrt{\frac{8d \log \frac{2d}{\delta}}{m}}.$$  

**Proof of Lemma 4.7.** The key observation in dealing with an unknown MDP is: either $\pi$, computed on the transition $\hat{P}$, is (almost) optimal for the given reward signal on the true MDP, or it escapes the set of known states $K$ quickly. If the former occurs, the requirement on the output of the algorithm is met. In case of the later, $\pi$ serves as a good policy to quickly explore new states – this can happen only a finite number of times.

Let $\pi^* = \arg \max_{\pi} V_{\pi}$. First, note that for any $\pi$ chosen in the Line 6, we have

$$V_{\pi} = \langle d_{\pi}, r \rangle \geq \langle d_{M_{K,\pi}}, r_K \rangle - (\gamma^t + P_{K}(\text{escape}|\pi))B$$

$$\geq \langle d_{M_{K,\pi}}, r_K \rangle - \frac{1}{1 - \gamma} \sqrt{\frac{8|S| \log \frac{2|S|}{\delta}}{m}} - (\gamma^t + P_{K}(\text{escape}|\pi))B$$

$$\geq \langle d_{M_{K,\pi}}, r_K \rangle - \varepsilon - \frac{1}{1 - \gamma} \sqrt{\frac{8|S| \log \frac{2|S|}{\delta}}{m}} - (\gamma^t + P_{K}(\text{escape}|\pi))B$$

$$\geq \langle d_{M_{K,\pi}}, r_K \rangle - \varepsilon - \frac{2}{1 - \gamma} \sqrt{\frac{8|S| \log \frac{2|S|}{\delta}}{m}} - (\gamma^t + P_{K}(\text{escape}|\pi))B$$

$$\geq V_{\pi^*} - \varepsilon - \frac{2}{1 - \gamma} \sqrt{\frac{8|S| \log \frac{2|S|}{\delta}}{m}} - (\gamma^t + P_{K}(\text{escape}|\pi))B$$

If $P_{K}(\text{escape}|\pi) > \Delta$, then the probability that $\pi$ doesn’t escape $K$ in $n$ trials is $e^{-n\Delta}$. Accounting for the failure probabilities with a suitable union bound, Line 8 ensures that $\pi$ is marked stable only if $P_K(\text{escape}|\pi) \leq \frac{\log(N\delta^{-1})}{n}$, where $N$ is the total number of times the inner loop is executed.

To observe the truth of the second part of the claim, note that every reiteration of the inner loop coincides with the exploration of some action at a $m$-unknown state. There can be at most $m|S||A|$ such exploration steps. Finally, each run of the inner loop samples $n$ episodes.
Proof of Lemma 4.8. First note that it suffices to ensure for all $t < t_0$ simultaneously, it happens $\|d_{t,\pi} - \hat{d}_{t,\pi}\|_{\infty} \leq 0.1 \varepsilon_0$. This is because

$$
\|d_{t,\pi} - \hat{d}_{t,\pi}\|_{\infty} \leq \frac{1 - \gamma}{(1 - \gamma t_0)} \sum_{t = 0}^{t_0 - 1} \gamma^t \|\hat{d}_{t,\pi} - (1 - \gamma t_0) d_{t,\pi}\|_{\infty} + \gamma^{t_0}
$$

$$
\leq \frac{1 - \gamma}{(1 - \gamma t_0)} \sum_{t = 0}^{t_0 - 1} \gamma^t \|\hat{d}_{t,\pi} - d_{t,\pi}\|_{\infty} + 0.3 \varepsilon_0 \leq \varepsilon_0.
$$

Since the trajectories are independently, $|\hat{d}_{t,\pi}(s) - d_{t,\pi}(s)| \leq \sqrt{\frac{2 \ln \frac{2}{\delta}}{n}}$ for each $t < t_0$ and state $s \in S$ with probability $1 - \delta$, by Hoeffding’s inequality. A union bound over states and $t$ concludes the proof.

5 Proof of Concept Experiments

We report the results from a preliminary set of experiments. In each case, the MaxEnt agent learns to access the set of reachable states within a small number of iterations, while monotonically increasing the entropy of the induced state distribution.

Recall that Algorithm 1 requires access to an approximate planning oracle and a density estimator for the induced distribution. Here the density estimator is deliberately chosen to be simple – a count-based estimate over the discretized state space. It is possible to use neural density estimators and other function-approximation based estimators in its stead.

\[\text{https://github.com/abbyvansoest/maxent_base} \text{ \text{and} \text{https://github.com/abbyvansoest/maxent_ant}}\]
Figure 3: The number of distinct \(xy\) states visited by \texttt{Ant} at various epochs. Results were averaged over \(N = 20\) executions. As the number of policies in the mixture increases, the agent reaches more unique states in the same amount of time.

5.1 Environments and the discretization procedure

The 2-dimensional state spaces for \texttt{MountainCar} and \texttt{Pendulum} (from \cite{BCP+16}) were discretized evenly to grids of size \(10 \times 9\) and \(8 \times 8\), respectively. For \texttt{Pendulum}, the maximum torque and velocity were capped at 1.0 and 7.0, respectively.

The 29-dimensional state space for \texttt{Ant} (with a Mujoco engine) was first reduced to dimension 7, combining the agent’s \(x\) and \(y\) location in the gridspace with a 5-dimensional random projection of the remaining 27 states. The \(x\) and \(y\) dimensions were discretized into 16 bins in the range \([-12, 12]\). The other dimensions were each normalized and discretized into 15 bins in the range \([-1, 1]\). While the planning agent agent had access to the full state representation, the density estimation was performed exclusively on the reduced representation.

5.2 Algorithmic details

\textbf{Reward function.} Each planning agent was trained to maximize a KL divergence objective function, ie. \(KL\text{(Unif} | d_x)\).

\texttt{MountainCar} and \texttt{Pendulum}. The planning oracle is a REINFORCE \cite{SMSM00} agent, where the the output policy from the previous iteration is used as the initial policy for the next iteration. The policy class is a neural net with a single hidden layer consisting of 128 units. The agent is trained on 400 and 200 episodes every epoch for \texttt{MountainCar} and \texttt{Pendulum}, respectively. The baseline agent chooses its action randomly at every time step.

\texttt{Ant}. The planning oracle is a Soft Actor-Critic \cite{HZAL18} agent. The policy class is a neural net with 2 hidden layers composed of 300 units and the ReLU activation function. The agent is trained for 30 episodes, each of which consists of a roll-out of 5000 steps. The mixed policy is executed over 10 trials of \(T = 10000\) steps at the end of each epoch in order to approximate the policy distribution and compute the next reward function. The baseline agent chooses its actions randomly for the same number of trials and steps.

\textbf{Acknowledgements}

Sham Kakade acknowledges funding from the Washington Research Foundation for Innovation in Data-intensive Discovery, the DARPA award FA8650-18-2-7836, and the ONR award N00014-18-1-2247. The authors thank Shie Mannor for helpful discussions.
References


