Provably Efficient Maximum Entropy Exploration

Elad Hazan [1 2] Sham M. Kakade [3 4 2] Karan Singh [1 2] Abby Van Soest [1 2]

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 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 the context of a given environment in the absence of 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[1] 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 (De Farias & Van Roy, 2003)) 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, ie. restricting the search to a non-linearly parameterized policy class (eg. 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-linearly parameterized policy classes. Given such an oracle, we give a provably efficient method for exploration based on the conditional gradient (or Frank-Wolfe) algorithm (Frank & Wolfe, 1956).

Formally, we show how to generate a sequence of reward signals, that when sequentially optimized give rise to a policy with an entropy on the state distribution 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 and that of the policy class. Next, we outline an efficient construction of such oracles and state the resultant sample & computational complexity in the tabular MDP setting. As a proof of concept, we implement the proposed method and demonstrate experiments over

[1]Department of Computer Science, Princeton University [2]Google AI Princeton [3]Allen School of Computer Science and Engineering, University of Washington [4]Department of Statistics, University of Washington. Correspondence to: Elad Hazan <ehazan@google.com>, Sham Kakade <sham@cs.washington.edu>, Karan Singh <karans@princeton.edu>, Abby Van Soest <asoest@princeton.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]In contrast to scalar rewards, such objectives permit a broader framework for reward specifiction, and may be useful in other contexts.

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 Π, we assume access to the environment using the following two oracles:

Approximate planning oracle: Given a reward function (on states) $r : \mathcal{S} \to \mathbb{R}$ and a sub-optimality gap ε, the planning oracle returns a policy $\pi = \textsc{ApproxPlan}(r, \varepsilon)$ with the guarantee that $V(\pi) \geq \max_{\pi \in \Pi} V(\pi) - \varepsilon$, where $V(\pi)$ is the value of policy π.

State distribution estimate oracle: A state distribution oracle estimates the state distribution $\hat{d}_\pi = \textsc{DensityEst}(\pi, \varepsilon)$ of any given (non-stationary) policy π, 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 β-smooth measure R, and any $\varepsilon > 0$, in $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ calls to* $\textsc{ApproxPlan}$ *&* $\textsc{DensityEst}$ *, it returns a policy $\bar{\pi}$ with*

$$R(d_{\bar{\pi}}) \geq \max_{\pi \in \Pi} R(d_\pi) - \varepsilon \,.$$

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. (Ng et al., 1999) where dense reward functions are provided to make the optimization problem more tractable) or through imitation learning (Abbeel & Ng, 2004; Ross et al., 2011) (where an expert demonstrates to the agent how to act).

PAC-RL 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 (Kearns & Singh, 2002; Kakade, 2003; Strehl et al., 2006; Lattimore & Hutter, 2014; Dann & Brunskill, 2015; Szita & Szepesvári, 2010; Azar et al., 2017). (Lim & Auer, 2012) offer a Dijkstra-like algorithm for discovering incrementally reachable states in the tabular setting.

Count-based Models & Directed Exploration: There are a host of recent empirical success using deep RL methods which encourage exploration in some form (Mnih et al., 2015; Silver et al., 2016). The approaches are based on a few related ideas: that of encouraging exploration through state visitation frequencies (e.g. (Ostrovski et al., 2017; Bellemare et al., 2016; Tang et al., 2017)) and those based on a intrinsic reward signal derived from novelty or prediction error (Lopes et al., 2012; Pathak et al., 2017; Savinov et al., 2018; Fu et al., 2017; Mohamed & Jimenez Rezende, 2015; Houthooft et al., 2016; Weber et al., 2017), aligning an intrinsic reward to the target objective (Kaelbling, 1993; Chentanez et al., 2005; Singh et al., 2010; 2009; Zheng et al., 2018), or sample based approaches to tracking of value function uncertainty (Osband et al., 2016; 2018).

Intrinsic Learning: Works in (Chentanez et al., 2005; Singh et al., 2009; 2010) 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) (Warde-Farley et al., 2018; Burda et al., 2018b;a; Nair et al., 2018). 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 (Chentanez et al., 2005; Singh et al., 2009; Warde-Farley et al., 2018; Nair et al., 2018)).

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 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 and actions (hereafter, referred to as a *trajectory*) is judged through the (discounted) cumulative reward it accumulates, defined as

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

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

$$V_\pi = \mathop{\mathbb{E}}_{\tau \sim P(\cdot|\pi)} V(\tau) = (1-\gamma) \mathop{\mathbb{E}}_{\tau \sim P(\cdot|\pi)} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

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

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

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

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

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

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

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

and the induced state distribution is:

$$d_{\pi_{\text{mix}}}(s) = \sum_{i=0}^{k-1} \alpha_i d_{\pi_i}(s). \tag{2.6}$$

Note that such a distribution over policies need not be representable as a stationary stochastic policy (even if the π_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 *concave* reward functional $R(\cdot)$ with this induced distribution. We say that a policy π^* is a *maximum-entropy exploration policy*, also to referred to as the *max-ent* policy, if the corresponding induced state distribution has the maximum possible $R(d_\pi)$ among the policy class Π. When considering the class of all policies, Lemma 3.3 assures us that the search over the class of stationary policies is sufficient.

$$\pi^* \in \arg\max_{\pi \in \Pi} R(d_\pi).$$

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[2].

$$\max_{\pi \in \Pi} \{H(d_\pi) = -\mathop{\mathbb{E}}_{s \sim d_\pi} \log d_\pi(s)\}$$

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

$$\min_{\pi \in \Pi} \left\{ \text{KL}(d_\pi || Q) = \mathop{\mathbb{E}}_{s \sim d_\pi} \log \frac{d_\pi(s)}{Q(s)} \right\}$$

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

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

where the expectation is now under Q. For uniform Q, this latter measure may be more aggressive in forcing π 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 *not* a concave function of the policy. Similar constructions can establish analogous statements for other non-trivial functionals. Subsequently, when the policy class in consideration is exhaustive, we discuss a possible convex reformulation of the objective in the space of induced distributions which constitute a convex set.

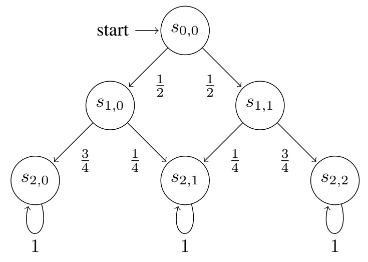
3.2.1. Non-convexity in the policy space

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.

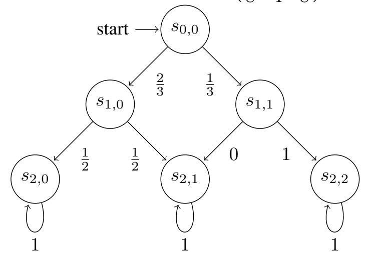
Lemma 3.1. *$H(d_\pi)$ is not concave in π.*

Proof. Figure 1 demonstrates the behavior of π_0, π_1, π_2 on a 6-state MDP with binary actions. Note that for sufficiently large $\gamma \to 1$ and any policy π, the discounted

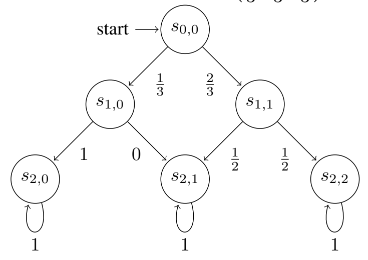
[2]Please note the distinction from the conditional entropy of actions given the state, e.g. (Todorov, 2007; Haarnoja et al., 2018).



(a) π_0, $d_{2,\pi_0} = \left(\frac{3}{8}, \frac{1}{4}, \frac{3}{8}\right)$



(b) π_1, $d_{2,\pi_1} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$



(c) π_2, $d_{2,\pi_2} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$

Figure 1. Description of π_0, π_1, π_2.

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 $\pi_0 = \frac{\pi_1+\pi_2}{2}$, observe that d_{2,π_0} is not uniform on $\{s_{2,0}, s_{2,1}, s_{2,2}\}$, implying that $H(d_{2,\pi_0}) < \frac{H(d_{2,\pi_1})+H(d_{2,\pi_2})}{2}$. □

Lemma 3.2. *For any policy π and MDP $\mathcal{M}$, define the matrix $P_\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ so that*

$$P_\pi(s', s) = \sum_{a\in\mathcal{A}} \pi(a|s)P(s'|s,a).$$

Then it is true that

1. *P_π is linear in π,*
2. *$d_{t,\pi} = P_\pi^t d_0$ for all $t \geq 0$,*
3. *$d_\pi = (1-\gamma)(I-\gamma P_\pi)^{-1}d_0$.*

Proof. Linearity of P_π 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 $\mathcal{K} = \{d : d(s,a) \geq 0$ and satisfies the constraints stated below$\}$. For every $d \in \mathcal{K}$, it is possible to construct a policy π with $d_\pi = d$, and for every π, $d_\pi \in \mathcal{K}$ holds (Puterman, 2014).

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

For an exhaustive policy class, the search for a max-ent policy can be recast as a convex optimization problem over the space of distributions.

$$\max_{d\in\mathcal{K}} R(d).$$

Although the above reduction is outlined for an exhaustive policy class, similar reductions are possible for linearly-parameterized policy classes (Peters et al., 2010; Neu et al., 2017). These techniques can be extended to the case of MDPs with unknown dynamics (De Farias & Van Roy, 2003).

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. *(Puterman, 2014) For any possibly non-Markovian policy π, define a stationary Markov policy π' as $\pi'(a|s) = \frac{d_\pi(s,a)}{d_\pi(s)}$. Then, $d_\pi = d_{\pi'}$.*

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) $r : \mathcal{S} \to \mathbb{R}$ and a sub-optimality gap ε_1, the planning oracle returns a policy[3] $\pi = \text{ApproxPlan}(r, \varepsilon_1)$ with the guarantee that $V_\pi \geq \max_{\pi\in\Pi} V_\pi - \varepsilon_1$.

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 π, guaranteeing that $\|d_\pi - \hat{d}_\pi\|_\infty \leq \varepsilon_0$.

[3]As the oracle is solving a discounted problem, we know the optimal value is achieved by a stationary policy.

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 $t = 0, \dots, T-1$ do
5:     Call the state distribution oracle on $\pi_{\text{mix},t} = (\alpha_t, C_t)$:
           $\hat{d}_{\pi_{\text{mix},t}} = \textsc{DensityEst}(\pi_{\text{mix},t}, \varepsilon_0)$
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 $r_t$:
           $\pi_{t+1} = \textsc{ApproxPlan}(r_t, \varepsilon_1)$ .
8:     Update $\pi_{\text{mix},t+1} = (\alpha_{t+1}, C_{t+1})$ to be
           $C_{t+1} = (\pi_0, \dots, \pi_t, \pi_{t+1})$,    (4.1)
           $\alpha_{t+1} = ((1-\eta)\alpha_t, \eta)$.    (4.2)
9: end for
10: return $\pi_{\text{mix},T} = (\alpha_T, C_T)$.
```

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

$$\|\nabla R(X) - \nabla R(Y)\|_\infty \le \beta \|X - Y\|_\infty \tag{4.3}$$

$$-\beta \mathbb{I} \preceq \nabla^2 R(X) \preceq \beta \mathbb{I}; \quad \|\nabla R(X)\|_\infty \le B \tag{4.4}$$

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:*

$$T \ge 10\beta\varepsilon^{-1} \log 10B\varepsilon^{-1} \,,$$

we have that:

$$R(d_{\pi_{mix,T}}) \ge \max_{\pi \in \Pi} R(d_\pi) - \varepsilon \,.$$

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_σ defined below.

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

When the algorithm is fed H_σ as the *proxy* reward functional, it is possible make sub-optimality guarantees on the true objective H. Lemma A.1 (D) relates the entropy functional H to its smoothed variant H_σ, while the rest of the lemma quantifies smoothness of H_σ. The factors of $|\mathcal{S}|$ 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 $\sigma = \frac{0.1\varepsilon}{2|\mathcal{S}|}$, $\varepsilon_1 = 0.1\varepsilon$, $\varepsilon_0 = \frac{0.1\varepsilon^2}{80|\mathcal{S}|}$, and $\eta = \frac{0.1\varepsilon^2}{40|\mathcal{S}|}$. When Algorithm 1 is run for T iterations with the reward functional H_σ, where:*

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

we have that:

$$H(d_{\pi_{mix,T}}) \ge \max_{\pi\in\Pi} H(d_\pi) - \varepsilon \,.$$

We continue with the proof of the main theorem.

Proof of Theorem 4.1. Let π^* be a maximum-entropy policy, ie. $\pi^* \in \arg\max_{\pi\in\Pi} R(d_\pi)$.

$$\begin{aligned} R(d_{\pi_{\text{mix},t+1}}) &= R((1-\eta)d_{\pi_{\text{mix},t}} + \eta d_{\pi_{t+1}}) && \text{Equation 2.6} \\ \ge & R(d_{\pi_{\text{mix},t}}) + \eta\langle d_{\pi_{t+1}} - d_{\pi_{\text{mix},t}}, \nabla R(d_{\pi_{\text{mix},t}})\rangle \\ & - \eta^2\beta\|d_{\pi_{t+1}} - d_{\pi_{\text{mix},t}}\|_2^2 && \text{smoothness} \end{aligned}$$

The second inequality follows from the smoothness[4] of R. To incorporate the error due to the two oracles, observe

$$\begin{aligned} &\langle d_{\pi_{t+1}}, \nabla R(d_{\pi_{\text{mix},t}})\rangle \\ &\ge \langle d_{\pi_{t+1}}, \nabla R(\hat{d}_{\pi_{\text{mix},t}})\rangle - \beta\|d_{\pi_{\text{mix},t}} - \hat{d}_{\pi_{\text{mix},t}}\|_\infty \\ &\ge \langle d_{\pi^*}, \nabla R(\hat{d}_{\pi_{\text{mix},t}})\rangle - \beta\varepsilon_0 - \varepsilon_1 \\ &\ge \langle d_{\pi^*}, \nabla R(d_{\pi_{\text{mix},t}})\rangle - 2\beta\varepsilon_0 - \varepsilon_1 \end{aligned}$$

The first and last inequalities invoke the assumptions laid out in Equation 4.3. Note that the second inequality above follows from the defining character of the planning oracle, ie. with respect to the reward vector $r_t = \nabla R(\hat{d}_{\pi_{\text{mix},t}})$, for any policy $\pi' \in \Pi$, it holds true that

$$V_{\pi_{t+1}} = \langle d_{\pi_{t+1}}, r_t\rangle \ge V_{\pi'} - \varepsilon_1 = \langle d_{\pi'}, r_t\rangle - \varepsilon_1$$

In particular, this statement holds[5] for the choice $\pi' = \pi^*$.

Using the above fact and continuing on

$$\begin{aligned} & R(d_{\pi_{\text{mix},t+1}}) \\ \ge & R(d_{\pi_{\text{mix},t}}) + \eta\langle d_{\pi^*} - d_{\pi_{\text{mix},t}}, \nabla R(d_{\pi_{\text{mix},t}})\rangle \\ & - 2\eta\beta\varepsilon_0 - \eta\varepsilon_1 - \eta^2\beta \\ \ge & (1-\eta)R(d_{\pi_{\text{mix},t}}) + \eta R(d_{\pi^*}) \\ & - 2\eta\beta\varepsilon_0 - \eta\varepsilon_1 - \eta^2\beta \end{aligned}$$

[4]See Section 2.1 in (Bubeck et al., 2015) for equivalent definitions of smoothness in terms of the function value and the Hessian.

[5]Even when when Π is chosen to be set of all stationary policies, this argument does not rely on π^* being a stationary policy, since π_{t+1} is an optimal policy for the reward function r_t among the class of all policies.

The last step here utilizes the concavity of R. Indeed, the inequality follows immediately from the sub-gradient characterization of concave functions. Now, with the aid of the above, we observe the following inequality.

$$\begin{aligned}&R(d_{\pi^*}) - R(d_{\pi_{\text{mix},t+1}})\\ &\leq (1-\eta)(R(d_{\pi^*}) - R(d_{\pi_{\text{mix},t}})) + 2\eta\beta\varepsilon_0 + \eta\varepsilon_1 + \eta^2\beta.\end{aligned}$$

Telescoping the inequality, this simplifies to

$$\begin{aligned}&R(d_{\pi^*}) - R(d_{\pi_{\text{mix},T}})\\ &\leq (1-\eta)^T(R(d_{\pi^*}) - R(d_{\pi_{\text{mix},0}})) + 2\beta\varepsilon_0 + \varepsilon_1 + \eta\beta\\ &\leq Be^{-T\eta} + 2\beta\varepsilon_0 + \varepsilon_1 + \eta\beta.\end{aligned}$$

Setting $\varepsilon_1 = 0.1\varepsilon$, $\varepsilon_0 = 0.1\beta^{-1}\varepsilon$, $\eta = 0.1\beta^{-1}\varepsilon$, $T = \eta^{-1}\log 10B\varepsilon^{-1}$ suffices. $\square$

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 (Bertsekas, 2005), (Fazel et al., 2018)), 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, where we consider the exhaustive class of policies.

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 (Bertsekas, 2005), eg. value iteration, linear programming. The state distribution oracle can be efficiently implemented as Lemma 3.2 suggests.

Corollary 4.3. *When the MDP $\mathcal{M}$ is known explicitly, with the oracles described in Section 4, Algorithm 1 runs in* $\text{poly}\left(\beta, |\mathcal{S}|, |\mathcal{A}|, \frac{1}{1-\gamma}, \frac{1}{\varepsilon}, \log B\right)$ *time to guarantee* $R(d_{\pi_{\mathit{mix},T}}) \geq \max_\pi R(d_\pi) - \varepsilon$.

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 γ-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 $\tilde{O}\left((1-\gamma)^{-1}\right)$-length episodes the agent must sample to achieve a ε-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 (Kearns & Singh, 2002). 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 4.4. *For an unknown MDP, with Algorithm 2 as the planning oracle and Algorithm 3 as the distribution estimate oracle, Algorithm 1 runs in* $\text{poly}\left(\beta, |\mathcal{S}|, |\mathcal{A}|, \frac{1}{1-\gamma}, \frac{1}{\varepsilon}\right)$ *time and executes* $\tilde{O}\left(\frac{B^3|\mathcal{S}|^2|\mathcal{A}|}{\varepsilon^3(1-\gamma)^2} + \frac{\beta^3}{\varepsilon^3}\right)$ *episodes of length* $\tilde{O}\left(\frac{\log|\mathcal{S}|\varepsilon^{-1}}{\log\gamma^{-1}}\right)$ *to guarantee that*

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

A sub-optimality bound may be derived on the non-smooth entropy functional H via Lemma A.1. Again, the extraneous factors introduced in the process are a consequence of the imposed smoothing via H_σ.

Corollary 4.5. *For an unknown MDP, with Algorithm 2 as the planning oracle and Algorithm 3 as the distribution estimate oracle and H_σ as the* proxy *reward functional, Algorithm 1 runs in* $\text{poly}\left(|\mathcal{S}|, |\mathcal{A}|, \frac{1}{1-\gamma}, \frac{1}{\varepsilon}\right)$ *time and executes* $\tilde{O}\left(\frac{|\mathcal{S}|^2|\mathcal{A}|}{\varepsilon^3(1-\gamma)^2} + \frac{|\mathcal{S}|^3}{\varepsilon^6}\right)$ *episodes of length* $\tilde{O}\left(\frac{\log|\mathcal{S}|\varepsilon^{-1}}{\log\gamma^{-1}}\right)$ *to guarantee that*

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

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 4.6. *For any reward function r with $\|r\|_\infty \leq B$, $\varepsilon > 0$, with $\varepsilon_1 = 0.1B^{-1}\varepsilon, m = \frac{32B^2|\mathcal{S}|\log\frac{2|\mathcal{S}|}{\delta}}{(1-\gamma)^2(0.1\varepsilon)^2}, n = \frac{B\log\frac{32|\mathcal{S}|^2|\mathcal{A}|\log\frac{2|\mathcal{S}|}{\delta}}{(1-\gamma)^2(0.1\varepsilon)^2\delta}}{0.1\varepsilon}, t_0 = \frac{\log\frac{0.1\varepsilon}{\log|\mathcal{S}|}}{\log\gamma}$, Algorithm 4.4 guarantees with probability $1-\delta$*

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

Furthermore, note that if Algorithm 4.4 is invoked T times (on possibly different reward functions), the total number

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 $C \in \mathbb{R}^{|\mathcal{S}|^2 \times |\mathcal{A}|}$, which is maintained across different calls to the planning algorithm to keep transition counts, to $C(s'|s,a) = 0$ for every $(s', s, a) \in \mathcal{S}^2 \times \mathcal{A}$.
3: repeat
4:     Declare $\mathcal{K} = \{s : \min_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} C(s'|s,a) \geq m\}$,
       $\hat{P}(s'|s,a) = \begin{cases} \frac{C(s'|s,a)}{\sum_{s'\in\mathcal{S}} C(s'|s,a)}, & \text{if } s \in \mathcal{K} \\ \mathbf{1}_{s'=s}. & \text{otherwise.} \end{cases}$
5:     Define the reward function as $r_{\mathcal{K}}(s) = \begin{cases} r(s), & \text{if } s \in \mathcal{K} \\ B. & \text{otherwise} \end{cases}$.
6:     Compute an (approximately) optimal policy on the MDP induced by $\hat{P}$ and reward $r_{\mathcal{K}}$. 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 $s \in \mathcal{S} - \mathcal{K}$, it chooses the least performed action.
       $\pi(s) = \begin{cases} (\Pi(r_{\mathcal{K}}, \varepsilon_1))(s) & \text{if } s \in \mathcal{K}, \\ \text{argmin}_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} C(s'|s,a) & \text{otherwise} \end{cases}$
7:     Run $\pi$ on the true MDP $\mathcal{M}$ to obtain $n$ independently sampled $t_0$-length trajectories $(\tau_1, \dots \tau_n)$, and increment the corresponding counts in $C(s'|s,a)$.
8:     If and only if no trajectory $\tau_i$ contains a state $s \in \mathcal{S} - \mathcal{K}$, mark $\pi$ as stable.
9: until $\pi$ is stable.
10: 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 $(\tau_0, \dots \tau_{m-1})$ of length $t_0$ following the policy $\pi$.
3: For every $t < t_0$, calculate the empirical state distribution $\hat{d}_{t,\pi}$.
       $d_{t,\pi}(s) = \frac{|\{i < m : \tau_i = (s_0, a_0, \dots) \text{ with } s_t = s\}|}{m}$
4: return $\hat{d}_\pi = \frac{1-\gamma}{1-\gamma^{t_0}} \sum_{t=0}^{t_0-1} \gamma^t \hat{d}_{t,\pi}$
```

of episodes sampled across all the invocations is $n(T + m|\mathcal{S}||\mathcal{A}|) = \tilde{O}\left(\frac{BT}{\varepsilon} + \frac{B^3|\mathcal{S}|^2|\mathcal{A}|}{\varepsilon^3(1-\gamma)^2}\right)$, *each episode being of length* t_0.

Lemma 4.7. *For any* $\varepsilon_0, \delta > 0$, *when Algorithm 3 is run with* $m = \frac{200}{\varepsilon_0^2} \log \frac{2|\mathcal{S}| \log 0.1\varepsilon}{\delta \log \gamma}$, $t_0 = \frac{\log 0.1\varepsilon_0}{\log \gamma}$, $\hat{d}_\pi$ *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 4.4. The claim follows immediately from the invocations of the two lemmas above with the parameter settings proposed in Theorem 4.1. □

5. Proof of Concept Experiments

We report the results from a preliminary set of experiments[6]. 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. In most of the experimental environments, 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, as we do for the `Humanoid` environment.

5.1. Environments and density estimation

Pendulum. The 2-dimensional state space for `Pendulum` (from (Brockman et al., 2016)) was discretized evenly to a grid of dimension 8×8. The dimensions represented angular position and angular velocity. For `Pendulum`, the maximum torque and velocity were capped at 1.0 and 7.0, respectively.

Ant. The 29-dimensional state space for `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.

Humanoid. The 376-dimensional state space for the Mujoco `Humanoid` environment was too large and complex to effectively discretize without significantly affecting the accuracy of estimation. In view of this, the agent's state space density was estimated using kernel density estimation (from scikit-learn (Pedregosa et al., 2011)), with an Epanechnikov kernel with a bandwidth of 0.10.

[6]The open-source implementations may be found at `https://github.com/abbyvansoest/maxent`.

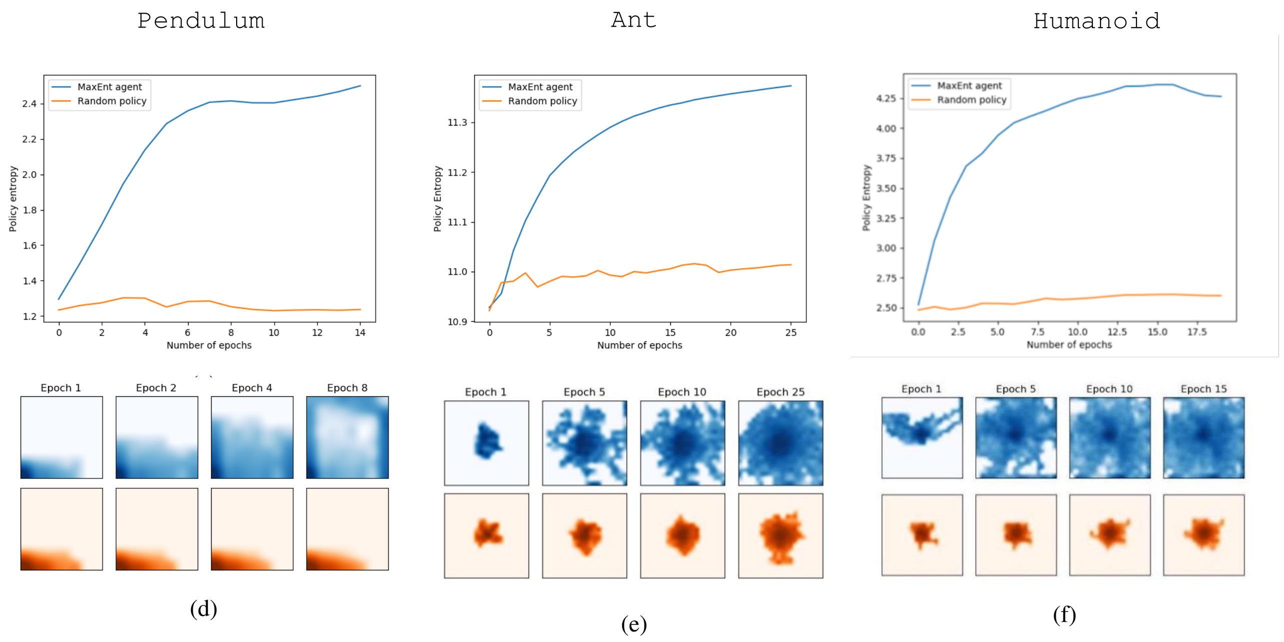


Figure 2. Results of the preliminary experiments. In each plot, blue represents the MaxEnt agent, and orange represents the random baseline. 2a, 2b, and 2c show the entropy of the policy evolving with the number of epochs. 2d, 2e, and 2f show the log-probability of occupancy of the two-dimensional state space. In 2e and 2f, the infinite xy grid is limited to $[-20, 20] \times [-20, 20]$.

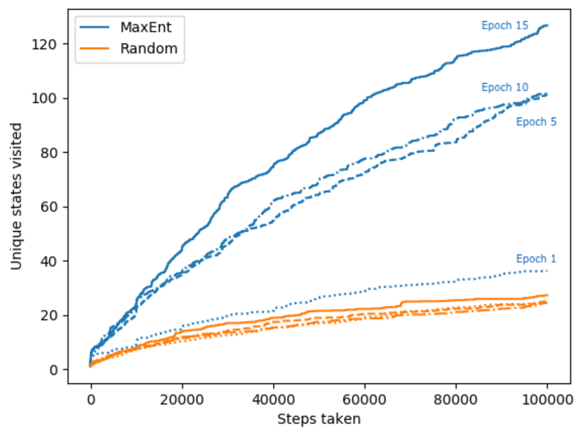


Figure 3. The number of distinct xy states visited by `Ant` at various epochs. Results were averaged over $N = 20$ executions. As the number of policies in the mixture increases, the agent reaches more unique states in the same amount of time.

5.2. Algorithmic details

Reward functional. Each planning agent was trained to maximize a smooth variant of the KL divergence objective.

$$\min_{\pi \in \Pi} \left\{ KL_\sigma(\text{Unif}||d_\pi) = - \mathop{\mathbb{E}}_{s\sim\text{Unif}} \log(d_\pi(s) + \sigma) + C \right\}.$$

Such a choice encourages visitation of novel states more aggressively than the entropy objective. The smoothing parameter was chosen to be $\sigma = |\mathcal{S}|^{-\frac{1}{2}}$.

Pendulum. The planning oracle is a REINFORCE (Sutton et al., 2000) 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 200 episodes every epoch. The baseline agent chooses its action randomly at every time step.

Ant. The planning oracle is a Soft Actor-Critic (Haarnoja et al., 2018) 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.

Humanoid. The planning oracle is a Soft Actor-Critic 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 for 20 trials of $T = 50,000$ steps to collect input data that is used to fit the kernel estimation model. This model is then used to estimate the induced state density of the mixed policy.

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

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. *arXiv preprint arXiv:1703.05449*, 2017.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1471–1479, 2016.

Bertsekas, D. P. *Dynamic programming and optimal control*, volume 1. Athena Scientific, 2005.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Bubeck, S. et al. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8 (3-4):231–357, 2015.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. In *arXiv:1808.04355*, 2018a.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Chentanez, N., Barto, A. G., and Singh, S. P. Intrinsically motivated reinforcement learning. In Saul, L. K., Weiss, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 17*, pp. 1281–1288. MIT Press, 2005.

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

De Farias, D. P. and Van Roy, B. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

Fazel, M., Ge, R., Kakade, S., and Mesbahi, M. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pp. 1466–1475, 2018.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956.

Fu, J., Co-Reyes, J., and Levine, S. Ex2: Exploration with exemplar models for deep reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2577–2587. Curran Associates, Inc., 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

Houthooft, R., Chen, X., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 1109–1117. Curran Associates, Inc., 2016.

Kaelbling, L. P. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambery, France, 1993. Morgan Kaufmann.

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

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

Lattimore, T. and Hutter, M. Near-optimal pac bounds for discounted mdps. *Theoretical Computer Science*, 558: 125–143, 2014.

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

Lopes, M., Lang, T., Toussaint, M., and yves Oudeyer, P. Exploration in model-based reinforcement learning by empirically estimating learning progress. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 206–214. Curran Associates, Inc., 2012.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Mohamed, S. and Jimenez Rezende, D. Variational information maximisation for intrinsically motivated reinforcement learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2125–2133. Curran Associates, Inc., 2015.

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. *CoRR*, abs/1807.04742, 2018.

Neu, G., Jonsson, A., and Gómez, V. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

Ng, A. Y., Harada, D., and Russell, S. J. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4026–4034. Curran Associates, Inc., 2016.

Osband, I., Aslanides, J., and Cassirer, A. Randomized prior functions for deep reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8625–8637. Curran Associates, Inc., 2018.

Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 2721–2730, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peters, J., Mulling, K., and Altun, Y. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T. P., and Gelly, S. Episodic curiosity through reachability. *CoRR*, abs/1810.02274, 2018.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Singh, S., Lewis, R. L., and Barto, A. G. Where do rewards come from? *Proceedings of the Annual Conference of the Cognitive Science Society (CogSci)*, 2009.

Singh, S. P., Lewis, R. L., Barto, A. G., and Sorg, J. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2:70–82, 2010.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888. ACM, 2006.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Szita, I. and Szepesvári, C. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 1031–1038, 2010.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. #exploration: A study of count-based exploration for deep reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2753–2762. Curran Associates, Inc., 2017.

Todorov, E. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pp. 1369–1376, 2007.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. *CoRR*, abs/1811.11359, 2018.

Weber, T., Racanière, S., Reichert, D. P., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.

Zheng, Z., Oh, J., and Singh, S. On learning intrinsic rewards for policy gradient methods. *CoRR*, abs/1804.06459, 2018.