Exploration by Maximizing Renyi Entropy for Reward-Free RL Framework

Chuheng Zhang*, **Yuanying Cai***, **Longbo Huang**, **Jian Li**
Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University
zhangchuheng123@live.com, cai-yy16@mails.tsinghua.edu.cn,
longbohuang@tsinghua.edu.cn, lijian83@mail.tsinghua.edu.cn

Abstract

Exploration is essential for reinforcement learning (RL). To face the challenges of exploration, we consider a reward-free RL framework that completely separates exploration from exploitation and brings new challenges for exploration algorithms. In the exploration phase, the agent learns an exploratory policy by interacting with a reward-free environment and collects a dataset of transitions by executing the policy. In the planning phase, the agent computes a good policy for any reward function based on the dataset without further interacting with the environment. This framework is suitable for the meta RL setting where there are many reward functions of interest. In the exploration phase, we propose to maximize the Rényi entropy over the state-action space and justify this objective theoretically. The success of using Rényi entropy as the objective results from its encouragement to explore the hard-to-reach state-actions. We further deduce a policy gradient formulation for this objective and design a practical exploration algorithm that can deal with complex environments. In the planning phase, we solve for good policies given arbitrary reward functions using a batch RL algorithm. Empirically, we show that our exploration algorithm is effective and sample efficient, and results in superior policies for arbitrary reward functions in the planning phase.

1 Introduction

The trade-off between exploration and exploitation is at the core of reinforcement learning (RL). Designing efficient exploration algorithm, while being a highly nontrivial task, is essential to the success in many RL tasks (Burda et al. 2019b; Ecoffet et al. 2019). Hence, it is natural to ask the following high-level question: *What can we achieve by pure exploration?* To address this question, several settings related to meta reinforcement learning (meta RL) have been proposed (see e.g., Wang et al. 2016; Duan et al. 2016; Finn, Abbeel, and Levine 2017). One common setting in meta RL is to learn a model in a reward-free environment in the meta-training phase, and use the learned model as the initialization to fast adapt for new tasks in the meta-testing phase (Eysenbach et al. 2019; Gupta et al. 2018; Nagabandi et al. 2019). Since the agent still needs to explore the environment under the new tasks in the meta-testing phase (sometimes it may need more new samples in some new task, and sometimes less), it is less clear how to evaluate the effectiveness of the exploration in the meta-training phase. Another setting is to learn a policy in a reward-free environment and test the policy under the task with a specific reward function (such as the score in Montezuma's Revenge) without further training with the task (Burda et al. 2019b; Ecoffet et al. 2019; Burda et al. 2019a). However, there is no guarantee that the algorithm has fully explored the transition dynamics of the environment unless we test the learned model for arbitrary reward functions. Recently, Jin et al. (2020) proposed *reward-free RL framework* that fully decouples exploration and exploitation. Further, they designed a provably efficient algorithm that conducts a finite number of steps of reward-free exploration and returns near-optimal policies for arbitrary reward functions. However, their algorithm is designed for the tabular case and can hardly be extended to continuous or high-dimensional state spaces since they construct a policy for each state.

The reward-free RL framework is as follows: First, a set of policies are trained to explore the dynamics of the reward-free environment in the exploration phase (i.e., the meta-training phase). Then, a dataset of trajectories is collected by executing the learned exploratory policies. In the planning phase (i.e., the meta-testing phase), an arbitrary reward function is specified and a batch RL algorithm (Lange, Gabel, and Riedmiller 2012; Fujimoto, Meger, and Precup 2019) is applied to solve for a good policy solely based on the dataset, without further interaction with the environment. This framework is suitable to the scenarios when there are many reward functions of interest or the reward is designed offline to elicit desired behavior. The key to success in this framework is to obtain a dataset with good coverage over all possible situations in the environment with as few samples as possible, which in turn requires the exploratory policy to fully explore the environment.

Several methods that encourage various forms of exploration have been developed in the reinforcement learning literature. The maximum entropy framework (Haarnoja et al. 2017) maximizes the cumulative reward in the meanwhile maximizing the entropy over *the action space* conditioned on each state. This framework results in several

*These authors contributed equally to this work. See the full version of this paper on arXiv:2006.06193.

efficient and robust algorithms, such as soft Q-learning (Schulman, Chen, and Abbeel 2017; Nachum et al. 2017), SAC (Haarnoja et al. 2018) and MPO (Abdolmaleki et al. 2018). On the other hand, maximizing *the state space* coverage may results in better exploration. Various kinds of objectives/regularizations are used for better exploration of the state space, including information-theoretic metrics (Houthooft et al. 2016; Mohamed and Rezende 2015; Eysenbach et al. 2019) (especially the entropy of the state space, e.g., Hazan et al. (2019); Islam et al. (2019)), the prediction error of a dynamical model (Burda et al. 2019a; Pathak et al. 2017; de Abril and Kanai 2018), the state visitation count (Burda et al. 2019b; Bellemare et al. 2016; Ostrovski et al. 2017) or others heuristic signals such as novelty (Lehman and Stanley 2011; Conti et al. 2018), surprise (Achiam and Sastry 2016) or curiosity (Schmidhuber 1991).

In practice, it is desirable to obtain a single exploratory policy instead of a set of policies whose cardinality may be as large as that of the state space (e.g., Jin et al. 2020; Misra et al. 2019). Our exploration algorithm outputs a single exploratory policy for the reward-free RL framework by maximizing *the Rényi entropy over the state-action space* in the exploration phase. In particular, we demonstrate the advantage of using state-action space, instead of the state space via a very simple example (see Section 3 and Figure 1). Moreover, Rényi entropy generalizes a family of entropies, including the commonly used Shannon entropy. We justify the use of Rényi entropy as the objective function theoretically by providing an upper bound on the number of samples in the dataset to ensure that a near-optimal policy is obtained for any reward function in the planning phase.

Further, we derive a gradient ascent update rule for maximizing the Rényi entropy over the state-action space. The derived update rule is similar to the vanilla policy gradient update with the reward replaced by a function of the discounted stationary state-action distribution of the current policy. We use variational autoencoder (VAE) (Kingma and Welling 2014) as the density model to estimate the distribution. The corresponding reward changes over iterations which makes it hard to accurately estimate a value function under the current reward. To address this issue, we propose to estimate a state value function using the off-policy data with the reward relabeled by the current density model. This enables us to efficiently update the policy in a stable way using PPO (Schulman et al. 2017). Afterwards, we collect a dataset by executing the learned policy. In the planning phase, when a reward function is specified, we augment the dataset with the rewards and use a batch RL algorithm, batch constrained deep Q-learning (BCQ) (Fujimoto, Meger, and Precup 2019; Fujimoto et al. 2019), to plan for a good policy under the reward function. We conduct experiments on several environments with discrete, continuous or high-dimensional state spaces. The experiment results indicate that our algorithm is effective, sample efficient and robust in the exploration phase, and achieves good performance under the reward-free RL framework.

Our contributions are summarized as follows:

- (Section 3) We consider the reward-free RL framework that separates exploration from exploitation and therefore places a higher requirement for an exploration algorithm. To efficiently explore under this framework, we propose a novel objective that maximizes the Rényi entropy over the state-action space in the exploration phase and justify this objective theoretically.
- (Section 4) We propose a practical algorithm based on a derived policy gradient formulation to maximize the Rényi entropy over the state-action space for the reward-free RL framework.
- (Section 5) We conduct a wide range of experiments and the results indicate that our algorithm is efficient and robust in the exploration phase and results in superior performance in the downstream planning phase for arbitrary reward functions.

2 Preliminary

A reward-free environment can be formulated as a controlled Markov process (CMP) [1] $(\mathcal{S}, \mathcal{A}, \mathbb{P}, \mu, \gamma)$ which specifies the state space $\mathcal{S}$ with $S = |\mathcal{S}|$, the action space $\mathcal{A}$ with $A = |\mathcal{A}|$, the transition dynamics $\mathbb{P}$, the initial state distribution $\mu \in \Delta^{\mathcal{S}}$ and the discount factor γ. A (stationary) policy $\pi_\theta : \mathcal{S} \to \Delta^{\mathcal{A}}$ parameterized by θ specifies the probability of choosing the actions on each state. The stationary discounted state visitation distribution (or simply the state distribution) under the policy π is defined as $d_\mu^\pi(s) := (1-\gamma)\sum_{t=0}^\infty \gamma^t \mathrm{Pr}(s_t = s | s_0 \sim \mu; \pi)$. The stationary discounted state-action visitation distribution (or simply the state-action distribution) under the policy π is defined as $d_\mu^\pi(s,a) := (1-\gamma)\sum_{t=0}^\infty \gamma^t \mathrm{Pr}(s_t = s, a_t = a | s_0 \sim \mu; \pi)$. Unless otherwise stated, we use d_μ^π to denote the state-action distribution. We also use d_{s_0,a_0}^π to denote the distribution started from the state-action pair (s_0, a_0) instead of the initial state distribution μ.

When a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is specified, CMP becomes the Markov decision process (MDP) (Sutton and Barto 2018) $(\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \mu, \gamma)$. The objective for MDP is to find a policy π_θ that maximizes the expected cumulative reward $J(\theta; r) := \mathbb{E}_{s_0\sim\mu}[V^{\pi_\theta}(s_0; r)]$, where $V^\pi(s; r) = \mathbb{E}[\sum_{t=0}^\infty \gamma^t r(s_t, a_t) | s_0 = s; \pi]$ is the state value function. We indicate the dependence on r to emphasize that there may be more than one reward function of interest. The policy gradient for this objective is $\nabla_\theta J(\theta; r) = \mathbb{E}_{(s,a)\sim d_\mu^{\pi_\theta}}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a; r)]$ (Williams 1992), where $Q^\pi(s, a; r) = \mathbb{E}[\sum_{t=0}^\infty \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a; \pi] = \frac{1}{1-\gamma}\langle d_{s,a}^\pi, r\rangle$ is the Q function.

Rényi entropy for a distribution $d \in \Delta^{\mathcal{X}}$ is defined as $H_\alpha(d) := \frac{1}{1-\alpha}\log\left(\sum_{x\in\mathcal{X}} d^\alpha(x)\right)$, where $d(x)$ is the probability mass or the probability density function on x (and the summation becomes integration in the latter case). When $\alpha \to 0$, Rényi entropy becomes Hartley entropy and equals the logarithm of the size of the support of d. When $\alpha \to 1$, Rényi entropy becomes Shannon entropy $H_1(d) :=$

[1] Although some literature use CMP to refer to MDP, we use CMP to denote a Markov process (Silver 2015) equipped with a control structure in this paper.

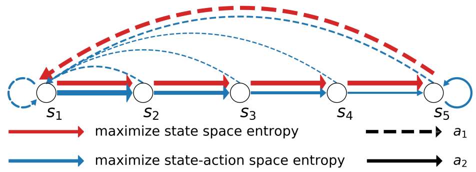

Figure 1: A deterministic five-state MDP with two actions. With discount factor $\gamma = 1$, a policy that maximizes the entropy of the discounted *state* visitation distribution does not visit all the transitions while a policy that maximizes the entropy of the discounted *state-action* visitation distribution visits all the transitions. Covering all the transitions is important for the reward-free RL framework. The width of the arrows indicates the visitation frequency under different policies.

$-\sum_{x\in\mathcal{X}} d(x)\log d(x)$ (Bromiley, Thacker, and Bouhova-Thacker 2004; Sanei, Borzadaran, and Amini 2016).

Given a distribution $d \in \Delta^{\mathcal{X}}$, the corresponding density model $P_\phi : \mathcal{X} \to \mathbb{R}$ parameterized by ϕ gives the probability density estimation of d based on the samples drawn from d. Variational auto-encoder (VAE) (Kingma and Welling 2014) is a popular density model which maximizes the variational lower bound (ELBO) of the log-likelihood. Specifically, VAE maximizes the lower bound of $\mathbb{E}_{x\sim d}[\log P_\phi(x)]$, i.e., $\max_\phi \mathbb{E}_{x\sim d, z\sim q_\phi(\cdot|x)}[\log p_\phi(x|z)] - \mathbb{E}_{x\sim d}[D_{\mathrm{KL}}(q_\phi(\cdot|x)||p(z))]$, where q_ϕ and p_ϕ are the decoder and the encoder respectively and $p(z)$ is a prior distribution for the latent variable z.

3 The Objective for the Exploration Phase

The objective for the exploration phase under the reward-free RL framework is to induce an *informative* and *compact* dataset: The informative condition is that the dataset should have good coverage such that the planning phase generates good policies for arbitrary reward functions. The compact condition is that the size of the dataset should be as small as possible to ensure a successful planning. In this section, we show that the Rényi entropy over the state-action space (i.e., $H_\alpha(d_\mu^\pi)$) is a good objective function for the exploration phase. We first demonstrate the advantage of maximizing the state-action space entropy over maximizing the state space entropy with a toy example. Then, we provide a motivation to use Rényi entropy by analyzing a deterministic setting. At last, we provide an upper bound on the number of samples needed in the dataset for a successful planning if we have access to a policy that maximizes the Rényi entropy over the state-action space. For ease of analysis, we assume the state-action space is discrete in this section and derive a practical algorithm that deals with continuous state-action space in the next section.

Why maximize the entropy over the state-action space? We demonstrate the advantage of maximizing the entropy over *the state-action space* with a toy example shown in Figure 1. The example contains an MDP with two actions and five states. The first action always drives the agent back to the first state while the second action moves the agent to the next state. For simplicity of presentation, we consider a case with a discount factor $\gamma = 1$, but other values are similar. The policy that maximizes the entropy of the state distribution is a deterministic policy that takes the actions shown in red. The dataset obtained by executing this policy is poor since the planning algorithm fails when, in the planning phase, a sparse reward is assigned to one of the state-action pairs that it visits with zero probability (e.g., a reward function $r(s, a)$ that equals to 1 on (s_2, a_1) and equals to 0 otherwise). In contrast, a policy that maximizes the entropy of the state-action distribution avoids the problem. For example, executing the policy that maximizes the Rényi entropy with $\alpha = 0.5$ over the state-action space, the expected size of the induced dataset is only 44 such that the dataset contains all the state-action pairs (cf. Appendix A). Note that, when the transition dynamics is known to be deterministic, a dataset containing all the state-action pairs is sufficient for the planning algorithm to obtain an optimal policy since the full transition dynamics is known.

Why using Rényi entropy? We analyze a deterministic setting where the transition dynamics is known to be deterministic. In this setting, the objective for the framework can be expressed as a specific objective function for the exploration phase. This objective function is hard to optimize but motivates us to use Rényi entropy as the surrogate.

We define $n := SA$ as the cardinality of the state-action space. Given an exploratory policy π, we assume the dataset is collected in a way such that the transitions in the dataset can be treated as i.i.d. samples from d_μ^π, where d_μ^π is the state-action distribution induced by the policy π.

In the deterministic setting, we can recover the exact transition dynamics of the environment using a dataset of transitions that contains all the n state-action pairs. Such a dataset leads to a successful planning for arbitrary reward functions, and therefore satisfies the *informative* condition. In order to obtain such a dataset that is also *compact*, we stop collecting samples if the dataset contains all the n pairs. Given the distribution $d_\mu^\pi = (d_1, \cdots, d_n) \in \Delta^n$ from which we collect samples, the average size of the dataset is $G(d_\mu^\pi)$, where

$$G(d_\mu^\pi) = \int_0^\infty \left(1 - \prod_{i=1}^n (1 - e^{-d_i t})\right) dt, \tag{1}$$

which is a result of the coupon collectors problem (Flajolet, Gardy, and Thimonier 1992). Accordingly, the objective for the exploration phase can be expressed as $\min_{\pi\in\Pi} G(d_\mu^\pi)$, where Π is the set of all the feasible policies. (Notice that due to the limitation of the transition dynamics, the feasible state-action distributions form only a subset of Δ^n.) We show the contour of this function on Δ^n in Figure 2a. We can see that, when any component of the distribution d_μ^π approaches zeros, $G(d_\mu^\pi)$ increases rapidly forming a *barrier* indicated by the black arrow. This barrier prevents the agent to visit a state-action with a vanishing probability, and thus encourages the agent to explore the hard-to-reach state-actions.

However, this function involves an improper integral which is hard to handle, and therefore cannot be directly used as an objective function in the exploration phase. One

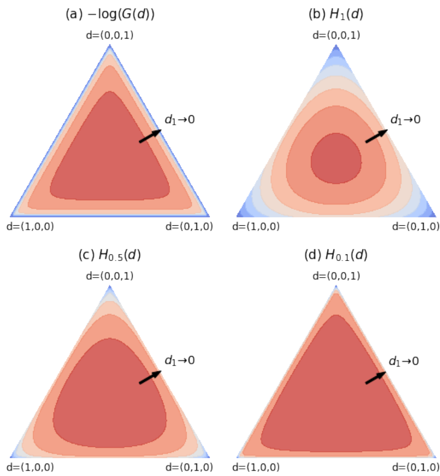

Figure 2: The contours of different objective functions on Δ^n with $n = 3$. $G(d)$ is the average size of the transition dataset sampled from the distribution d to ensure a successful planning. Compared with Shannon entropy (H_1), the contour of Rényi entropy ($H_{0.5}$, $H_{0.1}$) aligns better with G since Rényi entropy rapidly decreases when any component of d approaches zero (indicated by the arrow).

common choice for a tractable objective function is Shannon entropy, i.e., $\max_{\pi\in\Pi} H_1(d_\mu^\pi)$ (Hazan et al. 2019; Islam et al. 2019), whose contour is shown in Figure 2b. We can see that, Shannon entropy may still be large when some component of d_μ^π approaches zero (e.g., the black arrow indicates a case where the entropy is relatively large but $d_1 \to 0$). Therefore, maximizing Shannon entropy may result in a policy that visits some state-action pair with a vanishing probability. We find that the contour of Rényi entropy (shown in Figure 2c/d) aligns better with $G(d_\mu^\pi)$ and alleviates this problem. In general, the policy π that maximizes Rényi entropy may correspond to a smaller $G(d_\mu^\pi)$ than that resulted from maximizing Shannon entropy for different CMPs (more evidence of which can be found in Appendix B).

Theoretical justification for the objective function. Next, we formally justify the use of Rényi entropy over the state-action space with the following theorem. For ease of analysis, we consider a standard episodic setting: The MDP has a finite planning horizon H and stochastic dynamics $\mathbb{P}$ with the objective to maximize the cumulative reward $J(\pi; r) := \mathbb{E}_{s_1\sim\mu}[V^\pi(s_1; r)]$, where $V^\pi(s; r) := \mathbb{E}_{\mathbb{P}}\left[\sum_{h=1}^H r_h(s_h, a_h)|s_1 = s, \pi\right]$. We assume the reward function $r_h : \mathcal{S} \times \mathcal{A} \to [0, 1], \forall h \in [H]$ is deterministic. A (non-stationary, stochastic) policy $\pi : \mathcal{S} \times [H] \to \Delta^{\mathcal{A}}$ specifies the probability of choosing the actions on each state and on each step. The state-action distribution induced by π on the h-th step is $d_h^\pi(s, a) := \Pr(s_h = s, a_h = a|s_1 \sim \mu; \pi)$.

Theorem 3.1. *Denote ϖ as a set of policies $\{\pi^{(h)}\}_{h=1}^H$, where $\pi^{(h)} : S\times[H] \to \Delta^{\mathcal{A}}$ and $\pi^{(h)} \in arg\max_\pi H_\alpha(d_h^\pi)$. Construct a dataset $\mathcal{M}$ with M trajectories, each of which is collected by first uniformly randomly choosing a policy π from ϖ and then executing the policy π. Assume*

$$M \ge c\left(\frac{H^2SA}{\epsilon}\right)^{2(\beta+1)}\frac{H}{A}\log\left(\frac{SAH}{p\epsilon}\right),$$

where $\beta = \frac{\alpha}{2(1-\alpha)}$ and $c > 0$ is an absolute constant. Then, there exists a planning algorithm such that, for any reward function r, with probability at least $1 - p$, the output policy $\hat\pi$ of the planning algorithm based on $\mathcal{M}$ is 3ϵ-optimal, i.e., $J(\pi^*; r) - J(\hat\pi; r) \le 3\epsilon$, where $J(\pi^*; r) = \max_\pi J(\pi; r)$.

We provide the proof in Appendix C. The theorem justifies that Rényi entropy with small α is a proper objective function for the exploration phase since the number of samples needed to ensure a successful planning is bounded when α is small. When $\alpha \to 1$, the bound becomes infinity. The algorithm proposed by Jin et al. (2020) requires to sample $\tilde O(H^5S^2A/\epsilon^2)$ trajectories where $\tilde O$ hides a logarithmic factor, which matches our bound when $\alpha \to 0$. However, they construct a policy for each state on each step, whereas we only need H policies which easily adapts for the non-tabular case.

4 Method

In this section, we develop an algorithm for the non-tabular case. In the exploration phase, we update the policy π to maximize $H_\alpha(d_\mu^\pi)$. We first deduce a gradient ascent update rule which is similar to vanilla policy gradient with the reward replaced by a function of the state-action distribution of the current policy. Afterwards, we estimate the state-action distribution using VAE. We also estimate a value function and update the policy using PPO, which is more sample efficient and robust than vanilla policy gradient. Then, we obtain a dataset by collecting samples from the learned policy. In the planning phase, we use a popular batch RL algorithm, BCQ, to plan for a good policy when the reward function is specified. One may also use other batch RL algorithms. We show the pseudocode of the process in Algorithm 1, the details of which are described in the following paragraphs.

Policy gradient formulation. Let us first consider the gradient of the objective function $H_\alpha(d_\mu^\pi)$, where the policy π is approximated by a policy network with the parameters denoted as θ. We omit the dependency of π on θ. The gradient of the objective function is

$$\nabla_\theta H_\alpha(d_\mu^\pi) \propto \frac{\alpha}{1-\alpha}\mathbb{E}_{(s,a)\sim d_\mu^\pi}\Big[\nabla_\theta \log\pi(a|s) \left(\tfrac{1}{1-\gamma}\langle d_{s,a}^\pi, (d_\mu^\pi)^{\alpha-1}\rangle + (d_\mu^\pi(s,a))^{\alpha-1}\right)\Big]. \tag{2}$$

As a special case, when $\alpha \to 1$, the Rényi entropy becomes the Shannon entropy and the gradient turns into

$$\nabla_\theta H_1(d_\mu^\pi) = \mathbb{E}_{(s,a)\sim d_\mu^\pi}\Big[\nabla_\theta \log\pi(a|s) \left(\tfrac{1}{1-\gamma}\langle d_{s,a}^\pi, -\log d_\mu^\pi\rangle - \log d_\mu^\pi(s,a)\right)\Big]. \tag{3}$$

Due to space limit, the derivation can be found in Appendix D. There are two terms in the gradients. The first term

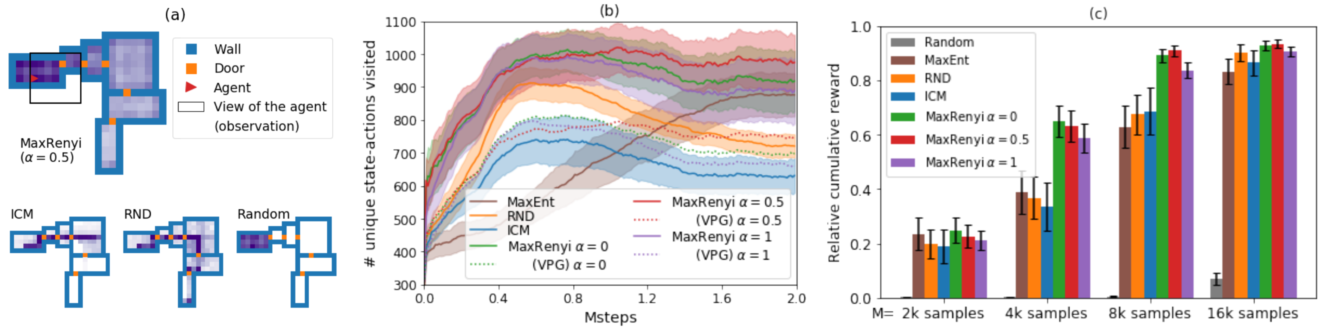

Figure 3: Experiments on `MultiRooms`. a) Illustration of the `MultiRooms` environment and the state visitation distribution of the policies learned by different algorithms. b) The number of unique state-action pairs visited by running the policy for 2k steps in each iteration in the exploration phase. c) The normalized cumulative reward of the policy computed in the planning phase (i.e., normalized by the corresponding optimal cumulative reward), averaged over different reward functions. The lines indicate the average across five independent runs and the shaded areas indicate the standard deviation.

equals to $\mathbb{E}_{(s,a)\sim d_\mu^\pi}[\nabla_\theta \log \pi(a|s) Q^\pi(s,a;r)]$ with the reward $r(s,a) = (d_\mu^\pi(s,a))^{\alpha-1}$ or $r(s,a) = -\log d_\mu^\pi(s,a)$, which resembles the policy gradient (of cumulative reward) for a standard MDP. This term encourages the policy to choose the actions that lead to the rarely visited state-action pairs. In a similar way, the second term resembles the policy gradient of instant reward $\mathbb{E}_{(s,a)\sim d_\mu^\pi}[\nabla_\theta \log \pi(a|s) r(s,a)]$ and encourages the agent to choose the actions that are rarely selected on the current step. The second term in (3) equals to $\nabla_\theta \mathbb{E}_{s\sim d_\mu^\pi}[H_1(\pi(\cdot|s))]$. For stability, we also replace the second term in (2) with $\nabla_\theta \mathbb{E}_{s\sim d_\mu^\pi}[H_1(\pi(\cdot|s))]$ [2]. Accordingly, we update the policy based on the following formula where $\eta > 0$ is a hyperparameter:

$$\nabla_\theta H_\alpha(d_\mu^\pi) \approx \begin{cases} \nabla_\theta J(\theta; r = (d_\mu^\pi)^{\alpha-1}) & 0 < \alpha < 1 \\ \nabla_\theta J(\theta; r = -\log d_\mu^\pi) & \alpha = 1 \end{cases} + \eta \nabla_\theta \mathbb{E}_{s\sim d_\mu^\pi}[H_1(\pi(\cdot|s))]. \quad (4)$$

Discussion. Islam et al. (2019) motivate the agent to explore by maximizing the Shannon entropy over the state space resulting in an intrinsic reward $r(s) = -\log d_\mu^\pi(s)$ which is similar to ours when $\alpha \to 1$. Bellemare et al. (2016) use an intrinsic reward $r(s,a) = \hat{N}(s,a)^{-1/2}$ where $\hat{N}(s,a)$ is an estimation of the visit count of (s,a). Our algorithm with $\alpha = 1/2$ induces a similar reward.

Sample collection. To estimate d_μ^π for calculating the underlying reward, we collect samples in the following way (cf. Line 5 in Algorithm 1): In the i-th iteration, we sample m trajectories. In each trajectory, we terminate the rollout on each step with probability $1-\gamma$. In this way, we obtain a set of trajectories $\mathcal{D}_i = \{(s_{j1}, a_{j1}), \cdots, (s_{jN_j}, a_{jN_j})\}_{j=1}^m$ where N_j is the length of the j-th trajectory. Then, we can use VAE to estimate d_μ^π based on $\mathcal{D}_i$, i.e., using ELBO as the density estimation (cf. Line 6 in Algorithm 1).

Value function. Instead of performing vanilla policy gradient, we update the policy using PPO which is more robust and sample efficient. However, the underlying reward function changes across iterations. This makes it hard to learn a value function incrementally that is used to reduce variance in PPO. We propose to train a value function network using relabeled off-policy data. In the i-th iteration, we maintain a replay buffer $\mathcal{D} = \mathcal{D}_i \cup \mathcal{D}_{i-1} \cup \cdots \cup \mathcal{D}_{i-n+1}$ that stores the trajectories of the last n iterations (cf. Line 5 in Algorithm 1). Next, we calculate the reward for each state-action pair in $\mathcal{D}$ with the latest density estimator P_ϕ, i.e., we assign $r = (P_\phi(s,a))^{\alpha-1}$ or $r = -\log P_\phi(s,a)$ for each $(s,a) \in \mathcal{D}$. Based on these rewards, we can estimate the target value $V^{\text{targ}}(s)$ for each state $s \in \mathcal{D}$ using the truncated TD(λ) estimator (Sutton and Barto 2018) which balances bias and variance (see the detail in Appendix F). Then,

[2] We found that using this term leads to more stable performance empirically since this term does not suffer from the high variance induced by the estimation of d_μ^π (see also Appendix E).

Algorithm 1 Maximizing the state-action space Rényi entropy for the reward-free RL framework

1: **Input:** The number of iterations in the exploration phase T; The size of the dataset M; The parameter of Rényi entropy α.
2: **Initialize:** Replay buffer that stores the samples form the last n iterations $\mathcal{D} = \emptyset$; Density model P_ϕ (VAE); Value function V_ψ; Policy network π_θ.
3: ▷ *Exploration phase* (MaxRenyi)
4: **for** $i = 1$ to T **do**
5: Sample $\mathcal{D}_i$ using π_θ and add it to $\mathcal{D}$
6: Update P_ϕ to estimate the state-action distribution based on $\mathcal{D}_i$
7: Update V_ψ to minimize $\mathcal{L}_\psi(\mathcal{D})$ defined in (5)
8: Update π_θ to minimize $\mathcal{L}_\theta(\mathcal{D}_i)$ defined in (6)
9: ▷ *Collect the dataset*
10: Rollout the exploratory policy π_θ, collect M trajectories and store them in $\mathcal{M}$
11: ▷ *Planning phase*
12: Reveal a reward function $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
13: Construct a labeled dataset $\overline{\mathcal{M}} := \{(s, a, r(s,a))\}$
14: Plan for a policy $\pi_r = \text{BCQ}(\overline{\mathcal{M}})$

we train the value function network V_ψ (where ψ is the parameter) to minimize the mean squared error w.r.t. the target values:

$$\mathcal{L}_\psi(\mathcal{D}) = \mathbb{E}_{s\sim\mathcal{D}}\left[(V_\psi(s) - V^{\text{targ}}(s))^2\right]. \quad (5)$$

Policy network. In each iteration, we update the policy network to maximize the following objective function that is used in PPO:

$$-\mathcal{L}_\theta(\mathcal{D}_i) = \mathbb{E}_{s,a\sim\mathcal{D}_i}\left[\min\left(\tfrac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}\hat{A}(s,a), g(\varepsilon, \hat{A}(s,a))\right)\right] + \eta\mathbb{E}_{s\sim\mathcal{D}_i}[H_1(\pi_\theta(\cdot|s))], \quad (6)$$

where $g(\varepsilon, A) = \begin{cases}(1+\varepsilon)A & A \geq 0\\ (1-\varepsilon)A & A < 0\end{cases}$ and $\hat{A}(s,a)$ is the advantage estimated using GAE (Schulman et al. 2016) and the learned value function V_ϕ.

5 Experiments

We first conduct experiments on the MultiRooms environment from minigrid (Chevalier-Boisvert, Willems, and Pal 2018) to compare the performance of MaxRenyi with several baseline exploration algorithms in the exploration phase and the downstream planning phase. We also compare the performance of MaxRenyi with different αs, and show the advantage of using a PPO based algorithm and maximizing the entropy over the state-action space by comparing with ablated versions. Then, we conduct experiments on a set of Atari (with image-based observations) (Machado et al. 2018) and Mujoco (Todorov, Erez, and Tassa 2012) tasks to show that our algorithm outperforms the baselines in complex environments for the reward-free RL framework. We also provide detailed results to investigate why our algorithm succeeds. More experiments and the detailed experiment settings/hyperparameters can be found in Appendix G.

Experiments on MultiRooms. The observation in MultiRooms is the first person perspective from the agent which is high-dimensional and partially observable, and the actions are turning left/right, moving forward and opening the door (cf. Figure 3a above). This environment is hard to explore for standard RL algorithms due to sparse reward. In the exploration phase, the agent has to navigate and open the doors to explore through a series of rooms. In the planning phase, we randomly assign a goal state-action pair, reward the agent upon this state-action, and then train a policy with this reward function. We compare our exploration algorithm MaxRenyi with ICM (Pathak et al. 2017), RND (Burda et al. 2019b) (which use different prediction errors as the intrinsic reward), MaxEnt (Hazan et al. 2019) (which maximizes the state space entropy) and an ablated version of our algorithm MaxRenyi(VPG) (that updates the policy directly by vanilla policy gradient using (2) and (3)).

For the exploration phase, we show the performance of different algorithms in Figure 3b. First, we see that variants of MaxRenyi performs better than the baseline algorithms. Specifically, MaxRenyi is more stable than ICM and RND that explores well at the start but later degenerates when the agent becomes familiar with all the states. Second, we observe that MaxRenyi performs better than MaxRenyi(VPG) with different αs, indicating that MaxRenyi benefits from adopting PPO which reduces variance with a value function and update the policy conservatively. Third, MaxRenyi with $\alpha = 0.5$ performs better than $\alpha = 1$ which is consistent with our theory. However, we observe that MaxRenyi with $\alpha = 0$ is more likely to be unstable empirically and results in slightly worse performance. At last, we show the visitation frequency of the exploratory policies resulted from different algorithms in Figure 3a. We see that MaxRenyi visits the states more uniformly compared with the baseline algorithms, especially the hard-to-reach states such as the corners of the rooms.

$M =$	2k	4k	8k	16k
MaxRenyi	**0.14**	**0.52**	**0.87**	**0.91**
State + bonus	**0.14**	0.48	0.78	0.87
State	0.07	0.15	0.41	0.57

Table 1: The lowest normalized cumulative reward of the planned policies across different reward functions, averaged over five runs. We use $\alpha = 0.5$ in the algorithms.

For the planning phase, we collect datasets of different sizes by executing different exploratory policies, and use the datasets to compute policies with different reward functions using BCQ. We show the performance of the resultant policies in Figure 3c. First, we observe that the datasets generated by running random policies do not lead to a successful planning, indicating the importance of learning a good exploratory policy in this framework. Second, the dataset with only 8k samples leads to a successful planning (with a normalized cumulative reward larger than 0.8) using MaxRenyi, whereas a dataset with 16k samples is needed to succeed in the planning phase when using ICM, RND or MaxEnt. This illustrates that MaxRenyi leads to a better performance in the planning phase (i.e., attains good policies with fewer samples) than the previous exploration algorithms.

Further, we compare MaxRenyi that maximizes the entropy over the state-action space with two ablated versions: "State + bonus" that maximizes the entropy over the state space with an entropy bonus (i.e., using P_ϕ to estimate the state distribution in Algorithm 1), and "State" that maximizes the entropy over the state space (i.e., additionally setting $\eta = 0$ in (6)). Notice that the reward-free RL framework requires that a good policy is obtained for arbitrary reward functions in the planning phase. Therefore, we show the lowest normalized cumulative reward of the planned policies across different reward functions for the algorithms in Table 1. We see that maximizing the entropy over the state-action space results in better performance for this framework.

Experiments on Atari and Mujoco. In the exploration phase, we run different exploration algorithms in the reward-free environment of Atari (Mujoco) for 200M (10M) steps and collect a dataset with 100M (5M) samples by executing the learned policy. In the planning phase, a policy is computed offline to maximize the performance under the extrinsic game reward using the batch RL algorithm based on the dataset. We show the performance of the planned policies in Table 2. We can see that MaxRenyi performs well on a range

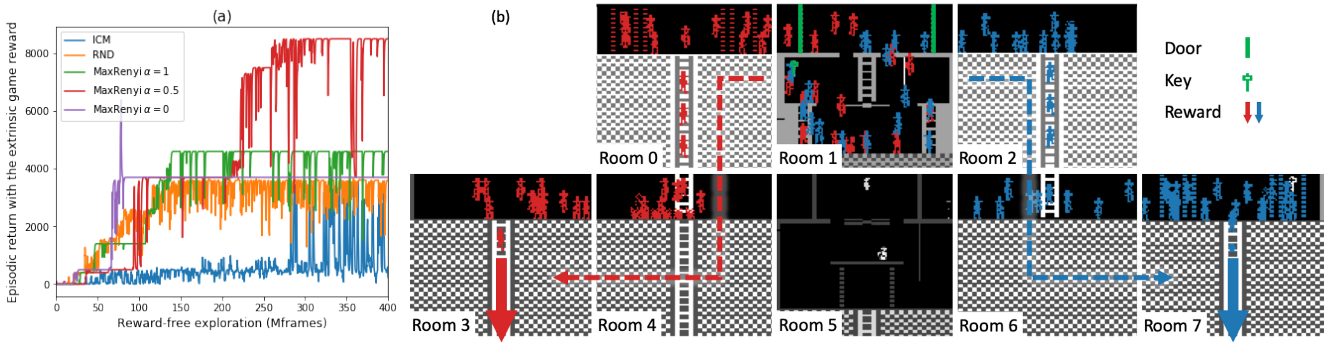

Figure 4: Experiments on Montezuma's Revenge. a) The episodic return with the extrinsic game reward along with the training in the exploration phase. b) The trajectory of the agent by executing the policy trained with the corresponding reward function in the planning phase. Best viewed in color.

	MaxRenyi	RND	ICM	Random
Montezuma	**8,100**	4,740	380	0
Venture	**900**	840	240	0
Gravitar	1,850	**1,890**	940	340
PrivateEye	**6,820**	3,040	100	100
Solaris	**2,192**	1,416	1,844	1,508
Hopper	**973**	819	704	78
HalfCheetah	**1,466**	1,083	700	54

Table 2: The performance of the planned policies under the reward-free RL framework corresponding to different exploratory algorithms, averaged over five runs. Random indicates using a random policy as the exploratory policy. We use $\alpha = 0.5$ in MaxRenyi.

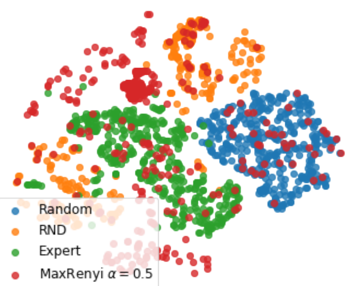

Figure 5: t-SNE plot of the state-actions randomly sampled from the trajectories generated by different policies in Hopper. Random policy chooses actions randomly. Expert policy is trained using PPO for 2M steps with extrinsic reward. RND and MaxRenyi policy are trained using corresponding algorithms without extrinsic reward.

of tasks with high-dimensional observations or continuous state-actions for the reward-free RL framework.

We also provide detailed results for Montezuma's Revenge and Hopper. For Montezuma's Revenge, we show the result for the exploration phase in Figure 4a. We observe that although MaxRenyi learns an exploratory policy in a reward-free environment, it achieves reasonable performance under the extrinsic game reward and performs better than RND and ICM. We also provide the number of visited rooms along the training in Appendix G, which demonstrates that our algorithm performs better in terms of the state space coverage as well. In the planning phase, we design two different sparse reward functions that only reward the agent if it goes through Room 3 or Room 7 (to the next room). We show the trajectories of the policies planned with the two reward functions in red and blue respectively in Figure 4b. We see that although the reward functions are sparse, the agent chooses the correct path (e.g., opening the correct door in Room 1 with the only key) and successfully reaches the specified room. This indicates that our algorithm generates good policies for different reward functions based on an offline planning in complex environments. For Hopper, we show the t-SNE (Maaten and Hinton 2008) plot of the state-actions randomly sampled from the trajectories generated by different policies in Figure 5. We can see that MaxRenyi generates state-actions that are distributed more uniformly than RND and overlap those from the expert policy. This indicates that the good performance of our algorithm in the planning phase is resulted from the better coverage of our exploratory policy.

6 Conclusion

In this paper, we consider a reward-free RL framework, which is useful when there are multiple reward functions of interest or when we design reward functions offline. In this framework, an exploratory policy is learned by interacting with a reward-free environment in the exploration phase and generates a dataset. In the planning phase, when the reward function is specified, a policy is computed offline to maximize the corresponding cumulative reward using the batch RL algorithm based on the dataset. We propose a novel objective function, the Rényi entropy over the state-action space, for the exploration phase. We theoretically justify this objective and design a practical algorithm to optimize for this objective. In the experiments, we show that our exploration algorithm is effective under this framework, while being more sample efficient and more robust than the previous exploration algorithms.

Acknowledgements

Chuheng Zhang and Jian Li are supported in part by the National Natural Science Foundation of China Grant 61822203, 61772297, 61632016, 61761146003, and the Zhongguancun Haihua Institute for Frontier Information Technology, Turing AI Institute of Nanjing and Xi'an Institute for Interdisciplinary Information Core Technology. Yuanying Cai and Longbo Huang are supported in part by the National Natural Science Foundation of China Grant 61672316, and the Zhongguancun Haihua Institute for Frontier Information Technology, China.

References

Abdolmaleki, A.; Springenberg, J. T.; Tassa, Y.; Munos, R.; Heess, N.; and Riedmiller, M. 2018. Maximum a posteriori policy optimisation. In *Proceedings of the 6th International Conference on Learning Representations*.

Achiam, J.; and Sastry, S. 2016. Surprise-based intrinsic motivation for deep reinforcement learning. In *Deep RL Workshop, Advances in Neural Information Processing Systems*.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 1471–1479.

Bromiley, P.; Thacker, N.; and Bouhova-Thacker, E. 2004. Shannon entropy, Renyi entropy, and information. *Statistics and Inf. Series (2004-004)* .

Burda, Y.; Edwards, H.; Pathak, D.; Storkey, A.; Darrell, T.; and Efros, A. A. 2019a. Large-scale study of curiosity-driven learning. In *Proceedings of the 7th International Conference on Learning Representations*.

Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2019b. Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations*.

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

Conti, E.; Madhavan, V.; Such, F. P.; Lehman, J.; Stanley, K.; and Clune, J. 2018. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, 5027–5038.

de Abril, I. M.; and Kanai, R. 2018. Curiosity-driven reinforcement learning with homeostatic regulation. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1–6. IEEE.

Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P. L.; Sutskever, I.; and Abbeel, P. 2016. RL2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* .

Ecoffet, A.; Huizinga, J.; Lehman, J.; Stanley, K. O.; and Clune, J. 2019. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995* .

Eysenbach, B.; Gupta, A.; Ibarz, J.; and Levine, S. 2019. Diversity is all you need: Learning skills without a reward function. In *Proceedings of the 7th International Conference on Learning Represention*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 1126–1135. JMLR.

Flajolet, P.; Gardy, D.; and Thimonier, L. 1992. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics* 39(3): 207–229.

Fujimoto, S.; Conti, E.; Ghavamzadeh, M.; and Pineau, J. 2019. Benchmarking Batch Deep Reinforcement Learning Algorithms. In *Deep RL Workshop, Advances in Neural Information Processing Systems*.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, 2052–2062. JMLR.

Gupta, A.; Eysenbach, B.; Finn, C.; and Levine, S. 2018. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640* .

Haarnoja, T.; Tang, H.; Abbeel, P.; and Levine, S. 2017. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 1352–1361. JMLR.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 1861–1870. JMLR.

Hazan, E.; Kakade, S.; Singh, K.; and Van Soest, A. 2019. Provably Efficient Maximum Entropy Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, 2681–2691. JMLR.

Houthooft, R.; Chen, X.; Duan, Y.; Schulman, J.; De Turck, F.; and Abbeel, P. 2016. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, 1109–1117.

Islam, R.; Seraj, R.; Bacon, P.-L.; and Precup, D. 2019. Entropy Regularization with Discounted Future State Distribution in Policy Gradient Methods. In *Optimization Foundations of RL Workshop, Advances in Neural Information Processing Systems*.

Jin, C.; Krishnamurthy, A.; Simchowitz, M.; and Yu, T. 2020. Reward-Free Exploration for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR.

Kingma, D. P.; and Welling, M. 2014. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*.

Lange, S.; Gabel, T.; and Riedmiller, M. 2012. Batch reinforcement learning. In *Reinforcement learning*, 45–73. Springer.

Lehman, J.; and Stanley, K. O. 2011. Novelty search and the problem with objectives. In *Genetic programming theory and practice IX*, 37–56. Springer.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9(Nov): 2579–2605.

Machado, M. C.; Bellemare, M. G.; Talvitie, E.; Veness, J.; Hausknecht, M. J.; and Bowling, M. 2018. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research* 61: 523–562.

Misra, D.; Henaff, M.; Krishnamurthy, A.; and Langford, J. 2019. Kinematic State Abstraction and Provably Efficient Rich-Observation Reinforcement Learning. *arXiv preprint arXiv:1911.05815* .

Mohamed, S.; and Rezende, D. J. 2015. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems*, 2125–2133.

Nachum, O.; Norouzi, M.; Xu, K.; and Schuurmans, D. 2017. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2775–2785.

Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R. S.; Abbeel, P.; Levine, S.; and Finn, C. 2019. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *Proceedings of the 7th International Conference on Learning Representations*.

Ostrovski, G.; Bellemare, M. G.; van den Oord, A.; and Munos, R. 2017. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2721–2730. JMLR. org.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 16–17.

Sanei, M.; Borzadaran, G. R. M.; and Amini, M. 2016. Renyi entropy in continuous case is not the limit of discrete case. *Mathematical Sciences and Applications E-Notes* 4.

Schmidhuber, J. 1991. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the international conference on simulation of adaptive behavior: From animals to animats*, 222–227.

Schulman, J.; Chen, X.; and Abbeel, P. 2017. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440* .

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2016. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the 4th International Conference of Learning Representations*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Silver, D. 2015. Lecture 2: Markov Decision Processes. *UCL. Retrieved from www0. cs. ucl. ac. uk/staff/d. silver/web/Teaching_files/MDP. pdf* .

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT press Cambridge.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* .

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4): 229–256.